VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

Diplomová práce

2011 Bc. David Říhošek

VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Distribuovaná prostředí pro testování síťových technologií

Distributed Environments for Testing of Networking Technologies

2011 Bc. David Říhošek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání diplomové práce

Student: **Bc. David Říhošek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Distribuovaná prostředí pro testování síťových technologií
Distributed Environments for Testing of Networking Technologies

Zásady pro vypracování:

Cílem práce je prozkoumat a popsat nejvýznamnější existující síťová prostředí vytvořená pro účely výzkumu, vývoje a testování síťových technologií. Konkrétním výstupem bude jejich srovnání a návrh nových vlastností pro systém Virtlab (http://www.virtlab.cz) vyvíjený na katedře informatiky FEI VŠB - TUO. Doporučována je orientace na systémy Federica, Emulab, PlanetLab, PanLab a případně další.

1. Vytipujte testovací prostředí pro bližší prozkoumání.
2. Zjistěte a popište architekturu a vlastnosti jednotlivých prostředí. Pokuste se jednotlivá prostředí také prakticky odzkoušet.
3. Seznamte se s infrastrukturními software tvořící základ jednotlivých prostředí a popište je.
4. Seznamte se s vlastnostmi systému Virtlab vyvíjeného na katedře informatiky FEI VŠB - TUO.
5. Na základě průzkumu vlastností existujících testovacích prostředí navrhněte perspektivní rozšíření systému Virtlab.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Petr Grygárek, Ph.D.**

Datum zadání: 19.11.2010
Datum odevzdání: 06.05.2011

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně.

Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 6.5.2011 v Ostravě.

Podpis

Poděkování

Rád bych poděkoval vedoucímu diplomové práce Ing. Petru Grygárkovi Ph.D. za odbornou pomoc a konzultace při vytváření této práce.

Abstrakt

Diplomová práce zpracovává problematiku testování síťových topologií pomocí testbedů. Obsahuje popis a příklady použití tří největších celosvětových testbedů. Dále jsou zde sepsány mé vlastní zkušenosti s použitím těchto systémů. Všechny systémy jsou rozebrány, jak po stránce praktického použití, tak po stránce jejich architektury. Také je v práci obsaženo srovnání s podobnými systémy vyvíjenými na VŠB- TUO. Na přiloženém CD jsou uvedeny okomentované praktické příklady z jednotlivých testbedů. Práce je určena jako rychlá příručka pro seznámení se s těmito systémy, obsahující návody a rady na správné a co nejefektivnější použití.

Klíčová slova

Emulab, Planetlab, Panlab, Federica, Virtlab, Testbed, Node, Slice, Resource, Teagle, Simulace, Virtuální stroj, Testování síťové topologie, RSpec, Používání testbedu, Práce s testbedem, VCT, VCT tool, Co Deploy, Implementace testbedu

Abstract

Diploma thesis process issues about testing network topologies with testbeds. It consists of describe and examples of usage of three worlds largest testbeds. There are also written in my own experience using these systems. All systems are analyzed both in terms of practical application, and in terms of their architecture. In conclusion the cake included a comparison with similar systems developing on VSB-TUO. On the CD are shown practical examples of annotated individual tested. The work is intended as a quick guide to familiarize yourself with these systems, which contains guidance and advice on the correct and most effective use.

Key words

Emulab, Planetlab, Panlab, Federica, Virtlab, Testbed, Node, Slice, Resource, Teagle, Simulation, Virtual Machine, Testing network topology, RSpec, Testbed usage, Work with testbed, VCT, VCT tool, Co Deploy, Testbed implementation

Seznam použitých symbolů a zkratek

DB	Databáze
DNS	Domain name server
IGW	Interconnection gateway
GUI	Grafické uživatelské rozhraní
HMAC	Keyed-hash Message Authentication Code
LAN	Local Area Network
MA	Management authority
NM	Node manager
NS-2	Network Simulator 2
PI	Project leader
PLC	Planetlab Consortium
PLL	Panlab
PTM	Panlab testbed manager
RAL	Resource adaptation
RSpec	Resource specification
SA	Slice authority
SW	Software
TCL	Tool command language
USRP	Universal software radio peripheral
VCT	Virtual
VM	Virtual machine
XML	Extensible Markup Language
XML RPC	XML Remote procedure call

Seznam obrázků

Obsah

1 Úvod

Diplomová práce je zaměřena na prozkoumání a otestování vybraných testbedů a jejich následné srovnání se systémy vyvíjenými na VŠB.

Práce je rozdělená do kapitol, kde každá se věnuje jednomu vybranému testbedu. Na začátku každé kapitoly je stručný úvod do systému, dále je popsáno architektonické řešení testbedu, následováno ukázkou nejpoužívanější syntaxe při tvorbě experimentů. Dále je v každé kapitole popsáno, jak se připojit k systému, koho je třeba kontaktovat, jak získat uživatelský účet a přístup do testbedu. V závěru každé kapitoly je uvedena sumace všech důležitých vlastností testbedu a porovnání se systémy vyvíjenými na VŠB. Nedílnou součástí každé kapitoly je také seznam pojmů, který popisuje obecné pojmy a zkratky používané při práci se systémem, aby čtenář měl přehled o názvech používaných v systému. Záměrně některé stěžejní pojmy v textu nejsou překládány, aby si čtenář zvykl na názvosloví používané přímo v systému. V závěru diplomové práce jsou všechny testované systémy na jednom místě srovnány a je zde uvedena sumarizace mých vlastních zkušeností s používáním těchto systémů. Celá práce je zaměřena na vlastní zkušenosti s testovanými testbedy. Diplomová práce je proto pojatá jako spojení teoretické části, která zahrnuje architekturu jednotlivých systémů a praktické části, která je zaměřena na provedení testovacího scénáře nad každým testbedem. Provedení tohoto scénáře je vždy komentováno osobními poznatky z používání systému. Dále jsou zde zdůrazněny důležité typy a triky, které usnadňují a urychlují práci se systémem. Celá práce je prolnuta mými vlastními zkušenostmi s používáním zvolených testovacích prostředí.

2 Emulab

Emulab je síťový testbed poskytující širokou škálu prostředí, ve kterých se dá sestavovat, ladit, testovat a hodnotit síťové systémy. Systém Emulab se skládá jak ze systémových zařízení, tak z podpůrného softwaru. Tento software se dá stáhnout ze stránek University of Utah. Zde je k dispozici obslužný software a odkazy na více než dvě desítky dalších stránek, které se zabývají problematikou testbedů. Jsou tam ukázky obsahující několik nebo i stovky uzlů. Emulab je využíván hlavně výzkumnými pracovníky z oblasti počítačových sítí a distribuovaných systémů. Je rovněž určen pro podporu vzdělávání v těchto oblastech.

Systém Emulab je přístupný přes webové rozhraní z adresy: http://www.emulab.net/ Před začátkem práce je třeba se samozřejmě zaregistrovat.

Emulab je veřejně přístupný a je k dispozici bezplatně pro výzkumné a vzdělávací účely. Pro bezplatný přístup k systému Emulab je potřeba splnit požadavky dokumentu *Administrative Policies* ze stránek: https://users.emulab.net/trac/emulab/wiki/AdminPolicies.

Přidělení účtu probíhá rychle a bez problémů. V mém případě bylo pouze potřeba potvrdit, že účet bude využíván pouze pro akademické a studijní účely.

Obrázek 1: Servery Emulabu, převzato z www.Emulab.com

2.1 Prostředí v Emulabu

Systém Emulab sjednocuje více různých prostředí[2] pod jedno uživatelské rozhraní a integruje je do jednotného rámce. Tento rámec poskytuje abstrakci služeb a jmenných prostorů stejnou pro všechny prostředí. Může se jednat například o pojmenování uzlů a vazeb, používání interní abstrakce[1] nebo používání vnitřních jmen.

Dále bude uveden seznam se stručným popisem jednotlivých prostředí[2] systému.

Emulation

Umožňuje pracovat na libovolné topologii sítě, která poskytuje kontrolovatelné, předpovídatelné a opakovatelné prostředí. Na koncových PC uzlech se dá pracovat jako uživatel *root* pod jakýmkoliv operačním systémem.

Live-Internet Experimentation

Pomocí testbedu PlametLab poskytuje Emulab plnohodnotné prostředí pro nasazení, testování a ladění aplikací na stovkách míst po celém světě. Existuje totiž možnost propojení těchto dvou testbedů a tvorby experimentů nad oběma systémy.

802.11 Wireless

Bezdrátové prostředí Emulabu 802.11a/b/g je umístěno v několika patrech kancelářské budovy. Veškeré uzly sítě jsou pod plnou kontrolou uživatele a mohou být nakonfigurovány jako přístupové body, klienti nebo v ad-hoc režimu. Všechny uzly mají dvě bezdrátová a jedno kabelové rozhraní.

Software-Defined Radio

Prostředí poskytující kontrolu nad první síťovou vrstvou bezdrátových sítí. Využívá zařízení USRP, pomocí kterých, se dá ovládat vše, počínaje zpracováním signálu pomocí softwaru.

Simulation

Toto prostředí umožňuje komunikovat emulovaným sítím, vytvořeným v simulátoru NS-2 s reálnými sítěmi, vytvořenými za pomoci prostředí Emulab.

[1] Interní abstrakce – Přístup k jednotlivým prostředím je sjednocen pod unifikované rozhraní. Prostředí jsou přístupné jak z vnějšku systému, tak zevnitř pomocí unifikovaného rozhraní.
[2] Prostředím, je zde myšlena část sytému, která poskytuje danou funkcionalitu.

2.2 Práce se systémem

V následující kapitole budou popsány mé zkušenosti s prací se systémem, bude zde uveden přehled funkcionality a návod, jak k systému přistupovat a používat jej.

2.2.1 Emulab network testbed software

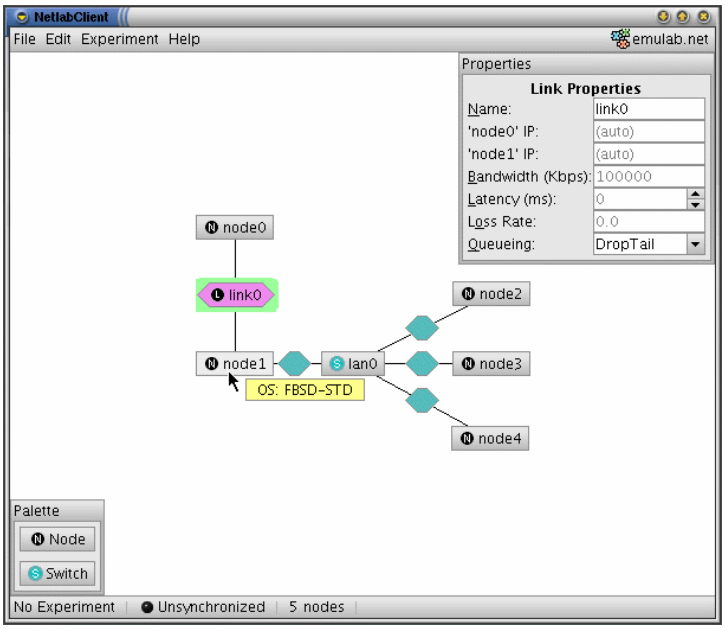
Software, *NetlabClient,* který si uživatel systému instaluje do svého počítače, a potom pomocí něj přistupuje k systému a tvoří své experimenty. V současnosti se používá verze 5.0, všechny ukázky budou proto prováděny v této verzi programu. Program je složen ze dvou částí: Výkonného jádra a GUI. Obě části se instalují samostatně.

Na webu je dostupná instalační příručka:

http://users.emulab.net/trac/emulab/wiki/InstallDocs

Instalace softwaru je jednoduchá, pokud se postupuje podle manuálu uvedeného výše, nenastanou žádné problémy. Instalace proběhne v řádu několika desítek vteřin. Na stroji 2 Ghz 4 GB RAM trvala přibližně 50 vteřin.

Po instalaci je software připraven k použití a navrhování testovaných topologií. Práce s programem je jednoduchá a rychlá. Pomocí GUI se dá rychle a přesně sestavit požadovaná topologie, ta se následně uloží do TCL skriptu a může být spuštěna buď přímo z menu programu nebo přes webové rozhraní systému. Navrhnout topologii složenou z deseti *nodes* propojených do LAN sítě, nastavit zpoždění linek a generování provozu trvá asi dvacet minut.



Obrázek 2: Netlab Client

2.2.2 Získání účtu na testbedu Emulab

Před začátkem práce se systémem, je třeba se prvně zaregistrovat. Registraci jednotlivých projektů provádí vedoucí projektu (pověřená osoba, např. vyučující, PI), ten potom k jeho jednotlivým projektům může přiřazovat spolupracující členy.

Jako první musí vedoucí projektu požádat o přidělení účtu Emulab na adrese:

http://www.emulab.net/reqaccount.php3

Potom záleží na správci systému, jestli tento projekt povolí nebo zamítne. Dále vedoucí přidělí práva jednotlivým členům týmu a může začít práce s testbedem. Při každém návratu na stránku testbedu nebo při práci s programem, je třeba se přihlásit.

V mém případě stačilo pouze do registračního formuláře napsat, na jaký účel budu testbed používat a správci systému ani nevyžadovali potvrzení od vyučujícího.

Při registraci a pro pozdější přihlašování je požadován SSH klíč. Dá se použít již stávající nebo si nechat vygenerovat nový, např. pomocí programu *PuTTY Key Generator*.

2.2.3 Přihlášení pomocí webového rozhraní

Pro používání systému je třeba mít vytvořený účet na stránkách www.emulab.net. Pro přístup k systému, je třeba na stránkách Emulabu zadat přihlašovací údaje a heslo. Jestli ještě nemáte vytvořen žádný projekt, je třeba také vybrat, jestli založit nový nebo se připojit ke stávajícímu. Dělá se to po přihlášení na hlavní stránce, pomocí tlačítek *Join Project* nebo *Start project*.

2.2.4 Vytváření nové topologie

První částí experimentu v systému Emulab je vytvoření potřebné topologie síťových zařízení, se kterými se potom bude pracovat. Pro tuto práci se dá využít buď GUI Java nebo sofistikovanější způsob pomocí skriptů ve formátu NS. Emulab používá skripty ve verzi NS-2. Jedná se o soubory, které detailně popisují síťovou topologii a interakci zařízení v čase, pomocí TCL skriptů. Více informací o psaní a syntaxi těchto skriptů, jako i o samotném simulátoru NS, je možné získat v mé bakalářské práci Simulátory počítačových sítí z roku 2009.

Protože však systém Emulab poskytuje spíše emulaci než simulaci síťových zařízení, některá funkcionalita NS skriptů může být málo odlišná nebo nemusí být plně implementována. To systém oznámí pomocí hlášky:

```
*** WARNING: Unsupported NS Statement!
```

Autoři systému se však snaží postupně doplňovat veškerou funkcionalitu. Kvůli tomu nemusí být všechny NS soubory spustitelné, jak v NS-2 simulátoru, tak v systému Emulab.

Pro příklad je v příloze diplomové práce uvedena jednoduchá okomentovaná topologie v NS souboru spustitelná jak v systému Emulab, tak v NS-2 simulátoru.

2.2.5 Spouštění experimentu

Po přihlášení do systému pomocí webového rozhraní, se vybere volba menu *Begin Experiment.* Napřed se zvolí, pod kterým projektem bude nový experiment probíhat, pokud je uživatel členem pouze jednoho projektu, nemá na výběr. Dalším krokem je vyplnění polí *Name* a *Description*. Do pole *Name* se zadává jedno slovo bez mezer, slouží pro pojmenování projektu, je možné použít oddělovače _. Do pole *description* se vyplní krátký popis projektu. Dále je potřeba vybrat umístění NS souboru na lokálním počítači, pomocí kterého se bude sestavovat požadovaná topologie. Pokračuje se tlačítkem *submit*. Tímto se spustí několikaminutový proces, během kterého se bude zadaný TCL skript zpracovávat. Délka zpracování závisí na počtu používaných zařízení, nastaveném zpoždění na linkách a přenosových rychlostech. Pokud vše proběhne bez problémů, systém odešle potvrzovací email, kde jsou informace o průběhu a úspěšnosti alokace požadovaných zařízení. Tento email bude obsahovat soupis všech požadovaných zařízení podle NS souboru a jejich IP adresy.

Ukázka potvrzovacího emailu:

```
Node Info:
 ID              Type         OS              Qualified Name
 --------------- ------------ --------- --------------------
 nodeA           pc                   nodeA.expediment.projekt.emulab.net
 nodeB           pc                   nodeB.expediment.projekt.emulab.net
 nodeC           pc                   nodeC.expediment.projekt.emulab.net
 nodeD           pc                   nodeD.expediment.projekt.emulab.net
 Physical Node Mapping:
 ID              Type         OS              Physical
 --------------- ------------ --------------- ------------
 tbsdelay0       pc850        FBSD47-STD      pc28
 nodeB           pc850        RHL73-STD       pc16
 nodeC           pc600        RHL73-STD       pc63
 nodeD           pc600        RHL73-STD       pc45
 nodeA           pc600        RHL73-STD       pc54
 Virtual Lan/Link Info:
 ID              Member          IP/Mask         Delay     BW (Kbs)  Loss Rate
 --------------- --------------- --------------- --------- --------- ---------
 lan0            nodeC:0         10.1.2.3        0.00      100000    0.000
                                 255.255.255.0   0.00      100000    0.000
 lan0            nodeB:1         10.1.2.4        0.00      100000    0.000
                                 255.255.255.0   0.00      100000    0.000
 lan0            nodeD:0         10.1.2.2        0.00      100000    0.000
                                 255.255.255.0   0.00      100000    0.000
 link0           nodeB:0         10.1.1.2        75.00     4000      0.005
                                 255.255.255.0   75.00     4000      0.004
 link0           nodeA:0         10.1.1.3        75.00     4000      0.005
                                 255.255.255.0   75.00     4000      0.007
 Physical Lan/Link Info:
 ID              Member          Delay Node   Delay    BW (Kbs) PLR    Pipe
 --------------- --------------- ------------ -------- -------- ------ ---------
 link0           nodeA           tbsdelay0    75.00    4000    0.250  100
 link0           nodeB           tbsdelay0    75.00    4000    0.100  110
 Route List:
 Node            Interface       Dest            Nexthop         Type  Cost
 --------------- --------------- --------------- --------------- ----- ----
 nodeA           10.1.1.3     10.1.2.4     10.1.1.2     host  0
 nodeA           10.1.1.3     10.1.2.0     10.1.1.2     net   0
 nodeC           10.1.2.3     10.1.1.0     10.1.2.4     net   0
 nodeD           10.1.2.2     10.1.1.0     10.1.2.4     net   0
```

2.2.6 Důležité poznámky pro sestavování topologie

Pro emulaci zpoždění na lince mezi *nodes A* a *B* se automaticky vytváří samostatný neviditelný *delay node*. Ten přidává na linku požadovanou latenci, chyby nebo snižuje přenosovou rychlost. Tyto parametry se nastavují buď v TCL skriptu nebo až po tvorbě experimentu.

Zpoždění méně než 2 ms na spoj jsou tak malé, že se nedají modelovat, tudíž je systém ignoruje. Zpoždění 0 ms se používá, pokud se ponechává původní "Fyzické" zpoždění spoje v systému.

Všechny spoje v sekci *Virtual Lan/Link* mají své zpoždění. Jedno zpoždění pro vstupující a jedno pro vystupující provoz. Z toho vyplývá, že zpoždění mez *nodes A* a *B* se vlastně skládá ze dvou virtuálních zpoždění, kvůli použití *virval node*.

Jména, ve sloupci *Qualified Name* slouží pro řídicí systém, který podle nich poznává, které *nodes* má alokované který uživatel. Tyto jména jsou předávána do *Emulab nameserver map* přímo při běhu expedrimentu. Slouží proto, aby uživatel nemusel znát fyzická jména nodes v systému, ale mohl pracovat pouze s vlastním zvoleným pojmenováním. O překlad jmen se stará *Emulab nameserver*. Dále jsou zde jména *myproj*, kde je uloženo jméno projektu a *myexp*, kam se ukládá název experimentu.

2.2.6.1 Použití nodes

Jakmile je uživateli odeslán potvrzovací email, jsou všechna požadovaná zařízení v systému nakonfigurována a připravena k použití. Pokud uživatel zvolil jeden z podporovaných operačních systémů pro *nodes*, systém nainstaloval a spustil požadovaný image systému na *node*. Tento konfigurační proces *node*, zahrnuje tyto části:

- Nahrání čistého obrazu disku, aby byl každý *node* vyčištěn od předchozích projektů.
- Restartování *node* a spuštění požadovaného operačního systému.
- Konfigurace požadovaných síťových rozhraní do stavu *up* a připojení do jejich *Lan (Vlan.)*
- Vytvoření uživatelských účtů pro všechny členy projektu.
- Nasdílení projektové složky do adresáře `/proj` pro lepší sdílení informací mezi *nodes* v rámci experimentu.
- Vytvoření a naplnění souboru `/etc/hosts` .
- Vytvoření a konfigurace všech *delay* parametrů.
- Konfigurace *serial console* pro přístup ke konzoli buď z users.emulab.net nebo přímo z plochy *node.*

Od této chvíle se uživatel může přihlásit na jakýkoliv ze zvolených *nodes* ve svém experimentu. K tomu se používá SSH a *qualified name* z *mapping table*. Je vidět, že se nemusí používat skutečná fyzická jména *nodes*. Přihlašovací jméno a heslo je stejné jako pro webové rozhraní systému. Soubor `/etc/hosts` na každém *node* poskytuje lokální mapování jmen pro uživatelský experiment. Pro komunikaci v rámci experimentu je třeba striktně používat mapování

uvedené v těchto souborech, a ne *.emulab.net names* uložené v *node mapping*. Tyto jména totiž slouží pro komunikaci na úrovni systému na *controll network*, která spravuje jednotlivé *nodes*.

Tento konfigurační proces platí pouze pro systémové image dodávané Emulabem. V systému však ještě existuje možnost, použít své vlastní image operačních systémů. V tomto případě je však uživatel zodpovědný sám za adresování a konfiguraci jednotlivých *nodes*.

2.2.6.2 Přístup jako Root, restart node

Pokud uživatel potřebuje měnit konfiguraci nebo restartovat *node*, dá se použít příkaz `sudo` uložený v `/usr/local/bin`. Pro FreeBSD a Linux nebo `/usr/pkg/bin` pro NetBSD. Pro systém Windows platí, že všichni uživatelé patří do skupiny *administrators*. Politika systému Emulab je v tomto velice liberální, může se měnit velké množství nastavení, ale musí se dodržovat pravidla politik *Emulab's administrative policies*. Pro příklad příkaz pro restart systému:

```
/usr/local/bin/sudo reboot
```

Pokud se rapidně sníží výkon některého *node*, kvůli přetížení procesoru nebo nastane jeho částečná nedostupnost, dají se *nodes* také restartovat dálkově, z `users.emulab.net` pomocí příkazu:

```
node_reboot <node> [node ... ]
```

Kde, *node,* je fyzické jméno node použité v *node mapping table*. Pomocí tohoto příkazu se také dá zadat více *nodes*. Nebo se také dá použít příkaz

```
node_reboot -e project,experiment
```

kde se restartují všechny *nodes* v požadovaném projektu a experimentu.

2.2.6.3 Přehrání image disku

Manuální přehrávání disku se nedoporučuje, existuje doporučený postup, kde se generuje celý nový experiment. Existuje však i možnost přehrát pouze jednotlivé obrazy disků na *nodes*. Všechna data, o které uživatel nechce přijít, by měl umístit do sdílené složky `/proj`. Přehrání obrazu disku totiž způsobí restart systému a kompletní smazání obsahu disku. Pro přehrání obrazu disku se používá příkaz

```
os_load <node> [node ... ]
```

Ten čeká, až budou všechny požadované obrazy přehrány a kontroluje správné provedení operace.

2.2.6.4 Ukončení experimentu

Jakmile uživatel dokončí svou práci na experimentu a dále nepotřebuje alokované zdroje, je potřeba experiment ukončit. Dělá se to přes webové rozhraní Emulabu pomocí odkazu *End An*

Experiment. Zde bude zobrazen seznam všech aktivních experimentů, ze všech projektů uživatele. Ukončení se provede klikem na tlačítko na pravé straně a potvrdí. Systém potom uvolní všechny alokovaná zařízení. Jakmile to dokončí, pošle uživateli potvrzovací email. V této chvíli se dá znova použít již použité jméno experimentu pro tvorbu nového experimentu, např. s jinými parametry.

Ukončení experimentu se také dá naplánovat pomocí *shedulleru*. To se zadává přímo, do konfiguračního TCL souboru, při vytváření konfigurace.

```
ns at 250.0 "$ns terminate"
```

Toho se dá využít při delších experimentech, v tomto příkladě, pokud bude emulovaná síť po 250 sekund neaktivní, systém automaticky provede ukončení experimentu.

2.3 Architektura systému

V této Kapitole budou popsány základní architektonické prvky systému, jejich implementace a opět několik praktických příkladů na ukázku.

2.3.1 Emulab testbed network

Každý fyzický node v testbedu má jedno rozhraní připojené do 100Mb LAN. Toto rozhraní je využíváno infrastrukturou testbedu k sestavování uživatelských experimentů. Dále se potom využívá uživatelem při experimentu pro komunikaci s ostatními *nodes*, nebo pro komunikaci ven ze systému Emulab. Nakonec je také používáno pro monitorování síťové aktivity, v průběhu experimentu. Toto fyzické připojení do sítě má svou pevnou IP adresu, kterou uživatel nemůže měnit. *Control network* [3]se tímto liší od *Experimental network*[4], kterou naopak uživatel používá při svých experimentech. Tato *Experimental network* se vždy vytváří virtuálně a slouží pouze pro jeden daný experiment. Veškeré spoje a zařízení v této síti jsou konfigurovány uživatelem.

Z místa mimo systém Emulab uživatel přistupuje k *nodes* v jeho experimentu pomocí Emulab jména stroje např. `pc126.emulab.net` nebo pomocí DNS, která se automaticky vytváří při každém experimentu, např. `node2.foo.testbed.emulab.net`. Oba druhy přístupu používají 100Mb *Control network* a přímý přístup pomocí Fyzických IP adres *nodes*.

Při pohledu "zevnitř " systému se adresování chová jinak. V ideálním případě, by se testovaná topologie měla tvářit, že neví nic o *Control network*. V praktických situacích, však musí

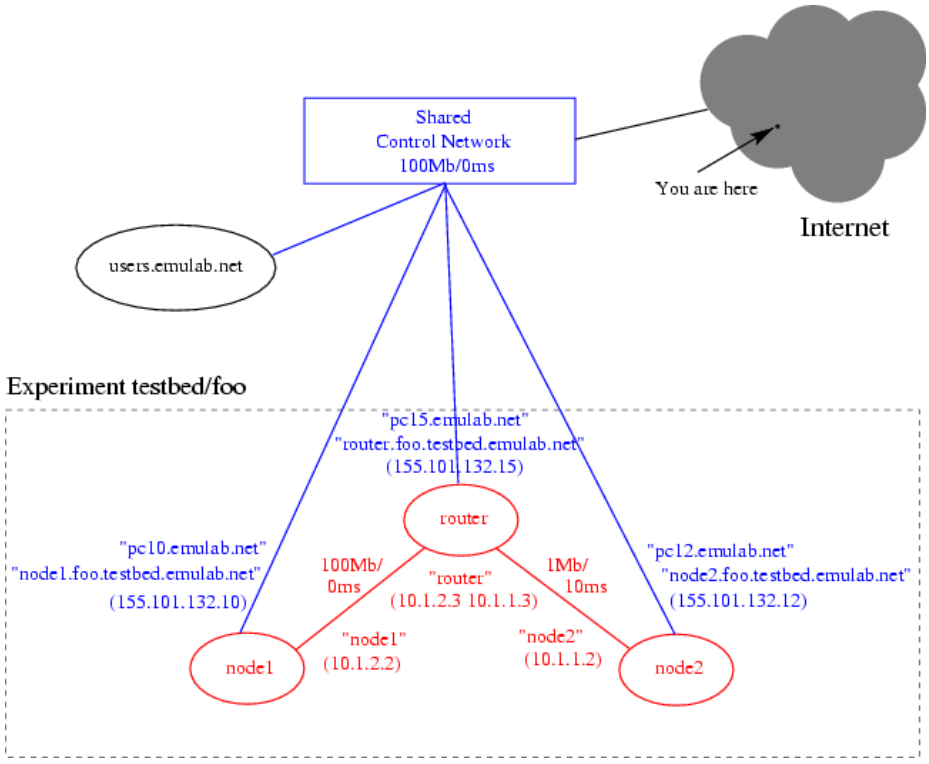
[3] Control network – Interní fyzická síť využívaná systémem pro interní komunikaci. Adresování je neměné.
[4] Experimental network – Uživatelská síť sestavuje se dynamicky podle jednotlivých experimentů. Využívá také fyzická zařízení.

nodes mít přístup do *Control network*. Využívá se toho při sdílení uživatelských složek a při přihlašování a ověřování uživatelů.

Je zde také rozdíl při směrování pokud se používají *qualified names*. Jména v experimentu jsou standardně překládána pomocí lokálního souboru `/etc/hosts`, ale *fully qualified names* jsou překládána pomocí *Emulab nameserver* a směrována do *controll network*.

Na ukázku rozdílu mezi těmito sítěmi a ukázky principu funkce je zde uveden příklad komunikace dvou PC mezi sebou přes jeden směrovač. Dále je také rozdíl graficky znázorněn na obrázku 3.

```
source tb_compat.tcl
set ns [new Simulator]
set node1 [$ns node]
set router [$ns node]
set node2 [$ns node]
set linkA [$ns duplex-link $node1 $router 100Mb 0ms DropTail]
set linkB [$ns duplex-link $router $node2 1Mb 10ms DropTail]
$ns rtproto Static
$ns run
```



Obrázek 3: Rozdíl mezi Control a User network, převzato z www.Emulab.net

2.3.2 Nastavení IP směrování

Díky tomu, že systém podporuje plnou uživatelskou kontrolu nad prostředím experimentu, není v experimentu vyžadován žádný směrovací protokol. Jednotlivé spoje mohou být adresovány manuálně. Další možností je použít některý z podporovaných směrovacích protokolů. Pro použití těchto protokolů se používá příkaz:

```
$ns rtproto ''protocol''
```

Jako protokol můžeme zvolit tyto možnosti: `Session`, `Static`, `Static-old`, nebo `Manual`.

`Session` – Směrování využívá OSPF směrovací protokol, který je spuštěn nad všemi nodes. Tento protokol není podporován obrazem operačního systému Win XP.

`Static` – Jedná se o automatické směrování, kdy jsou směrovací tabulky sestavovány dynamicky, na každém node zvlášť.

`Static-old` – Jedná se o centralizované směrování, kdy jsou při zavedení sestaveny směrovací tabulky a jsou nahrány, do jednotlivých *node*, ve chvíli, kdy *nodes* startují.

`Manual` - Jedná se o manuální nastavování směrovacích tabulek na každém *node,* pomocí příkazu `nexthop`.

```
$node add-route $dst $nexthop
```

Kde, `dst` , může být node, linka nebo síť viz příklad.

```
$client add-route $server $router
$client add-route [$ns link $server $router] $router
$client add-route $serverlan $router
```

2.4 Rozhraní pro systém PlanetLab

Systém Emulab poskytuje komponentu, pomocí které je možné propojit tyto dva testbedy a získat tak možnost sestavovat topologie v celosvětovém měřítku.

Z praktického hlediska je vhodné použití *nodes* ze systému PlanetLab tehdy, pokud je třeba pracovat nad skutečnými sítěmi, jejich zpožděními chybovostmi atd. Například při výběru *nodes* je možné zvolit několik *nodes* z opačných konců světa a pracovat tak nad reálným prostřením Internetu.

2.4.1 Tvorba slice PlanetLab

Před použitím tohoto rozhraní, je třeba mít vytvořený účet v systému PlanetLab. Další věcí, které je potřeba, je předpřipravit si v systému PlanetLab nový *slice*, do kterého se potom budou přidávat požadované *nodes*. To už se děje přes rozhraní v systému Emulab. Z vlastních zkušeností vím, že je daleko rychlejší připojovat se pomocí systému Emulab k již předpřipravenému *slice* a ten potom pouze nastavit a přidat si do něj požadované *nodes*.

Další možností je vytvořit si *slice* přímo přes systém Emulab. Dělá se to ve standardním programu, určeném pro ovládání systému Emulab, *Netlab Client*. Pro tvorbu *slice* se jednoduše používá menu *Create a (PlanetLab) Slice*. Dále je potřeba vyplnit požadované údaje týkající se výběru *slice* a nastavení jeho parametrů. Jedná se především o jméno *slice* a jeho popis. Dále se vybere *submit*, tím se potvrdí tvorba nového *slice*. Dále je třeba vyčkat na potvrzovací email, který potvrdí tvorbu *slice*. V mém případě to trvalo jeden den, než přišel. Proto je lepší si vytvořit *slice* přímo v systému PlanetLab a potom se k němu pouze připojit. Je to mnohem rychlejší. Detailní postup tvorby *slice* v systému Planetlab, je popsán v kapitole věnované tomuto systému.

2.4.2 Použití PlanetLab Slice

Jakmile je nový *slice* vytvořen, dá se prohlédnout v menu *Experiment List* nebo v menu na webových stránkách systému Emulab v sekci *'My Testbed/Emulab' page*. Tato stránka také umožňuje provádět různá nastavení *slice*. Dále se zde dá provádět změna stavů *slice* a mazání. Dále je zde zobrazeno, kolik a které *nodes* jsou aktuálně ve *slice* použity a také se dají prohlížet statistiky jejich aktivity. Samozřejmostí je také nastavení a správa jednotlivých *nodes*.

Použití těchto *nodes* v experimentu v systému Emulab je jednoduché. Pro přístup a použití *nodes* se využívá syntaxe psané přímo ve vstupním TCL skriptu pro systém Emulab. Pro adresování *nodes* se využívá DNS jmen ve formátu:

`node>.<exp>.<proj>.emulab.net`

Kde `node` je jméno node v rámci *slice.*

`exp` je jméno experimentu v systému Panlab, jedná se o stejné jméno jako jméno *slice*.

`proj` je jméno aktuálního projektu v systému Emulab.

Použití *nodes* ze systému PlanetLab se dále řídí stejnou syntaxí, jako použití standardních *nodes* v systému Emulab.

Bohužel, majitelé systému Emulab uvažují o ukončení podpory tohoto rozhraní pro komunikaci mezi oběma systémy. V polovině dubna 2011 bylo rozhraní plně funkční a zatím nebyl oznámen termín ukončení podpory.

2.5 Závěr

Systém Emulab sice pracuje nad fyzickými počítači, *nodes*, ale využívá je pouze jako koncové stanice. Další síťová rozhraní jako jsou směrovače, přepínače nebo dokonce i celé LAN sítě jsou na těchto fyzických strojích pouze virtualizovány. Ve skutečnosti to znamená, že pokud uživatel požaduje vytvořit jednoduchou síťovou topologii, ve které budou dva počítače navzájem propojené přes směrovač, systém Emulab k tomu využije tři *nodes*. Na dvou *nodes* bude spuštěn požadovaný virtuální obraz operačního systému a ten se bude starat o komunikaci uvnitř experimentu. Ve skutečnosti na stanici běží jiný operační systém a ten komunikuje s *controll network* na úrovni správy projektů. V jednu chvíli, na jednom *node,* může běžet pouze jeden virtuální stroj simulující skutečný operační systém. O simulaci směrovače, se bude starat, třetí *node,* na tom bude spuštěn také obraz operačního systému, ale ten se bude starat pouze o směrování komunikace mezi ostatními *nodes*. Bude se tvářit jako směrovač, ale ve skutečnosti to bude pouze počítač, na kterém je spuštěn virtuální obraz směrovače.

Systém Emulab je primárně určen pro testování síťových technologií, ne sítí samotných, proto poskytuje většinu možností na straně klientské stanice, poskytuje několik možných operačních systémů a dává možnost instalovat a spouštět uživatelský software. Dále poskytuje velké možnosti při nastavování linek mezi zařízeními, nastavení zpoždění, chybovosti atd. Kvůli svému zaměření spíše na klientskou část sítí jsou možnosti simulace jiných síťových zařízení, než koncových uzlů značně omezené.

Pokud by se měly navzájem porovnat systémy Virtlab a Emulab, bylo by na první pohled jasné, že každý systém je zaměřen na jiný typ uživatelů. Zatímco Virtlab se snaží věrně simulovat všechna síťová zařízení a jejich nastavení a dělá to pomocí virtuálního propojování skutečných síťových prvků, systém Emulab mezitím slouží spíše pro testování různých síťových aplikací a jejich chování v reálných podmínkách větších sítí. Z tohoto důvodu zde postačují menší možnosti nastavování v oblasti směrovacích a přepínacích zařízení, které jsou naopak vyváženy věrnou simulací veškeré funkcionality na straně klientských stanic.

Systém Virtlab by mohl ze systému Emulab čerpat nápad, na použití univerzálních skriptů, které může napřed uživatel vyzkoušet “nanečisto“ v simulačním programu NS-2 nebo jiném softwaru, kde by se dala překontrolovat jejich správnost a až potom, co je bude mít otestované, je spouštět na fyzických zařízeních. Použití simulátoru NS- 2 by sice v případě systému Virtlab nebylo vhodné, protože pracuje na stejné filozofii jako systém Emulab. Je zaměřen spíše na komunikaci koncových stanic, než na samotnou topologii a směrovací a přepínací zařízení. Pro ukládání sestavovaných topologií by bylo vhodné využít TCL skripty, jako u systému Emulab nebo XML dokumenty, pomocí kterých by se navzájem předávaly data jak o požadovaných síťových topologiích, tak by do nich mohl systém například generovat výsledky experimentů a ty by potom mohly být vizualizovaný nezávisle na platformě a systému, který bude uživatel používat.

3 Federica

Federica je evropský projekt určený na vývoj nových internetových technologií. Začal v roce 2008 a ukončení proběhlo v červnu 2010. Nejedná se o klasický testbed, kde by si jednotliví uživatelé mohli testovat své návrhy topologií nebo nových technologií. Testbed Federica je určen pro testování nových protokolů, služeb a aplikací. Byli zde zapojeni velcí partneři jako např. Cesnet, UPC, NORDUnet a mnoho dalších.

Projekt Federica skončil v roce 2010. A protože byl určen pouze pro velké partnery, jak již bylo popsáno výše, nepodařilo se mi k němu připojit a získat uživatelský účet. Protože se jedná o již ukončený projekt, nebude zde popsána ani jeho architektura, ani jeho příklad použití.

4 Planetlab

Planetlab je celosvětová výzkumná síť, která se snaží podporovat výzkum v oblasti vývoje síťových služeb. Od začátku roku 2003, kdy byla založena, už sdružuje více než 1 000 výzkumníků akademických pracovníků a průmyslových výzkumných laboratoří, kteří ji využívají k výzkumu nových technologií pro distribuovaná úložiště dat, mapování sítí, peer to peer systémy, distribuované hash tabulky a technologie front. V současné době Planetlab obsahuje 1133 uzlů ve všech částech světa a kolem pěti set aktivních projektů, viz http://www.planetlab.org. Mezi hlavní spolupracovníky patří např. firmy HP, Intel nebo France Telecom. Z akademické sféry je možné jmenovat např. University of California at Berkeley, Princeton University a University of Washington.

Žádný stát ani žádná univerzita nebudou mít nikdy dostatečné prostředky, aby vytvořili dostatečně velkou laboratoř pro výzkum síťových služeb. Díky tomu, že na projektu Planetlab spolupracuje takové množství různých autorit, má tento projekt velké ambice, stát se celosvětově největší laboratoří pro vývoj a výzkum síťových služeb.

Mezi hlavní výhody Planetlabu v oblasti testování síťových technologií patří to, že jednotlivé uzly jsou rozloženy po celé planetě, což umožňuje testování v reálném prostředí celosvětového Internetu.

4.1.1 Aktivity CESNETu v rámci sítě Planetlab

CESNET se připojil do celosvětové laboratoře Planetlab v červenci 2006, do evropské laboratoře Planetlab.eu potom v roce 2009. Od té doby umožňuje všem českým akademickým pracovištím přístup do této celosvětové síťové laboratoře.

Primárním cílem této angažovanosti, bylo poskytnout české akademické komunitě možnost spolupráce s předními světovými odborníky na problematiku síťové komunikace.

4.1.2 Principy vztahů v systému

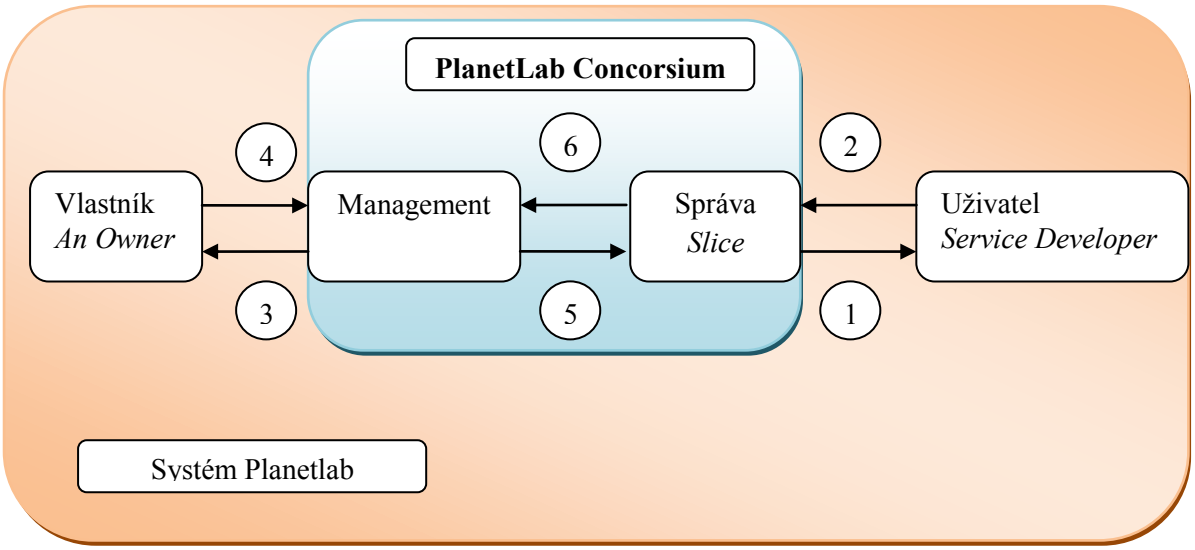
V systému PlanetLab vystupují tři hlavní skupiny účastníků.

Vlastník (An Owner) Jedná se o organizaci, která poskytuje jeden nebo více *nodes*. Každý vlastník si ponechává hlavní kontrolu nad svými vlastními *node*s, ale poskytuje je pod správu důvěryhodného zprostředkovatele služeb PLC – PlanetLab Concorsium. PLC poskytuje vlastníkům *nodes* mechanizmy pro aplikaci bezpečnostních politik na jejich vlastní *node*.

Uživatel (A User) Je výzkumník, který využívá služeb nad sadou *nodes* v síti PlanetLab. Uživatelé systému jsou v současné době omezeni na osoby zaměstnané důvěryhodnými výzkumnými organizacemi. Uživatelé mohou vytvářet *slice* nad skupinami *nodes* v síti PlanetLab pomocí mechanizmů stanovených autoritou PLC.

PlanetLab Consorcium (PLC) je důvěryhodný zprostředkovatel služeb, který spravuje jednotlivé *nodes*, které jsou pod dohledem jejich vlastníků, vytváří uživateli požadované *slice* nad skupinami požadovaných *nodes*.

Na následujícím obrázku jsou přehledně znázorněny vztahy v rámci systému PlanetLab.



Obrázek 4: Principy vztahů v systému

1. PLC předpokládá, že se jedná o důvěryhodného, ověřeného uživatele. Ověřování v systému je zajištěno pomocí SSH klíčů. Ověřený uživatel musí patřit do nějaké důvěryhodné organizace.
2. Uživatel důvěřuje PLC, ta jedná jeho jménem, vytváří *slice* na požadovaných *nodes*, takže uživatel může instalovat a upravovat software běžící na svém *slice*.
3. Vlastníci důvěřují PLC, ta instaluje a spravuje software pro monitorování síťové aktivity, zajišťuje optimální spojení *slices* a *nodes* a zajišťuje správu a využití jednotlivých *nodes*.
4. PLC věří vlastníkům, že jejich *nodes* jsou fyzicky bezpečné, PLC zajišťuje bezpečnostní politiky. PLC musí také ověřit, že každý *node*, který spravuje, patří vlastníkovi, s nímž má uzavřenou smlouvu.
5. Řídící orgán zajišťuje autenticitu spojení mezi *slice* a *nodes*. Management také spravuje bezpečnostní politiky a *blacklist* s uživateli a *slices*, kteří nesmí využívat určité služby.
6. Každý správce *slice* a uživatelé, jejímž jménem jedná, spadají pod určitý řídící orgán. Řídící orgán zajišťuje autenticitu pracovního prostředí, v němž uživatel pracuje.

4.2 Slovník pojmů

Node – Jednotlivý fyzický počítač připojený do systému Planetlab. Nejedná se o počítač, na kterém pracuje uživatel, ale o počítač, ke kterému se připojuje a provádí na něm testování svých aplikací. Někdy se jim také může říkat Fyzické zdroje. Může být také nazýván uzel.

Slice – Pracovní projekt, který sdružuje několik uživatelů a několik uzlů.

PI – Člověk z nadřazené instituce, který potvrzuje registraci a dohlíží na práci v systému.

Host key – Přesné jednoznačné označení node v systému PlanetLab. Je dostupné z internetu na adrese: https://www.planet-lab.org/db/nodes/known_hosts.php

PlanetLab Concorsiun – PLC, správní orgán systému Planetlab, dohlíží na funkčnost systému a řeší problémy se systémem.

4.3 Architektura systému

Tato kapitola popisuje základní architektonické prvky systému. Tyto prvky se skládají ze dvou základních částí.

1. Sada softwarových modulů, které fungují v každém uzlu.
2. Kolekce globálních mechanizmů, které implementuje autorita PLC.

Vždy je uveden jak český překlad, tak anglický název, se kterým se pracuje v systému.

4.3.1 Uzel (Node)

Node, je fyzický počítač připojený do internetu, který je schopen hostovat jeden nebo i více virtuálních strojů (VM). *Node*, musí mít alespoň jednu veřejnou IP adresu. Každý *node* je identifikován pomocí unikátního ID, které je přímo vázané na sadu atributů určených výslovně pro tento *node*. Např. IP adresa, MAC adresa a hostující organizace. S výjimkou prefixu adresy sítě, do které tento *node,* patří, mohou být tyto atributy měněny. Toto umožňuje účinnou výměnu HW, na kterém *node*, běží. A to buď postupně nebo v celku. Ale uzly (jejich ID) nemohou migrovat z jednoho místa do druhého. IP adresa uzlu nemusí být univerzálně směrovatelná, ale musí být dosažitelná z místa, kde běží komponenty PLC, tj. musí být dosažitelná z internetu pro všechny komponenty PLC. V současné době musí být IP adresa uzlu přidělena staticky, ale v brzké budoucnosti bude moci být toto prováděno pomocí DHCP.

V současnosti se používá většinou mapování jednoho VM na jeden fyzický počítač, ale není to podmínkou.

Není požadavkem, aby všechny uzly měly stejnou architekturu, ale současná implementace je omezena pouze na procesory x86, protože v systému zatím žádné jiné nejsou použity. Architektura také předpokládá, že bude existovat prostředek, který umožní PLC vzdálený restart *node*. Většinou se používá *HP Lights-Out,* ale není to podmínkou. Dalším požadavkem je možnost záznamu výstupu na konzoli.

Uzel startuje za pomoci dvou komponent, které musejí být dostupné offline. Jedná se o soubory `bootfile` a `plnoode.txt`, kde jsou uloženy síťová nastavení, spolu s veřejným klíčem pro připojení k PLC. Tyto musí být při startu dostupné a za jejich dostupnost zodpovídá vlastník.

`Bootfile` je spustitelný obraz, který obsahuje aktuální konfiguraci uzlu. Obsahuje taktéž pokyny pro kontaktování PLC pro stažení všech potřebných softwarových komponent.

`Plnode.txt` je textový soubor unikátní pro každý uzel. Obsahuje hlavní síťová nastavení. V následujícím příkladu je uveden obsah souboru.

```
IP ADDRESS = „128.145.10.147“
HOST NAME = „planetlab1.cs.foo.edu“
NET DEV = „00:04:75:26:14:ee“
NODE KEY = „145eabd81547896542de60477....“
NODE ID = „205“
```

V současnosti jsou soubory dostupné přes CD nebo USB, v budoucnu budou kompilovány do jednoho samostatného USB zařízení nebo dostupné přes síť.

V současné době systém na všech *nodes* používá operační systém Fedora Core 2. Každý *node*, obsahuje čistou instalaci systému, bez jakýchkoli aplikací. Ty si musí uživatel na *node* nahrát sám. *Node* se používají výhradně pro testování uživatelských aplikací, jejich vývoj musí proběhnout ještě před nahráním na *node*.

4.3.2 Virtuální stroj (Virtual Machine, VM)

Virtuální stroj je softwarové prostředí implementované na *node*, ve kterém běží jednotlivé *slice*. Na jednom *node* může běžet vedle sebe více virtuálních strojů. Virtuální stroj je implementován pomocí NM popsaného v následující kapitole.

1. Jednotlivé virtuální stroje musí být od sebe izolovány
    - Zdroje spotřebovávané jedním virtuálním strojem, nesmí ovlivnit výkonnost druhého virtuálního stroje. Zdroje jsou rozdělovány spravedlivě.
    - Žádný virtuální stroj nemůže odposlouchávat síťový provoz jiného virtuálního stroje.
    - Žádný virtuální stroj nemůže získat přístup k objektům, datům nebo procesům jiného virtuálního stroje.
2. Uživatelé mají možnost zabezpečeného vzdáleného přihlášení do virtuálního stroje, pomocí svých přihlašovacích údajů.
3. Virtuální stroj provede všechny uživatelské *boot* skripty při každém startu.
4. Uživatelé mohou instalovat softwarové balíčky ve virtuálním stroji bez dopadu na ostatní balíčky běžící v jiných virtuálních strojích na témže uzlu.
5. Odchozí provoz z Virtuálního stroje může být monitorován za účelem bezpečnosti. Monitoruje se: čas, doba trvání, zdrojová a cílová adresa a port a uživatel.
6. Pro přístup k jinému virtuálnímu stroji je potřeba souhlas tohoto virtuálního stroje.

Ve skutečnosti se používá mapování 1:1 mapování fyzického a virtuálního stroje. Protože však systém umožňuje dynamické odpojování neaktivních *slices*, může na jednom fyzickém *node* běžet i více *slices*, ale pouze jeden bude aktivní a bude se s ním pracovat.

4.3.3 Manager uzlu (Node Manager NM)

Node manager je program, který běží na každém *node,* který vytváří virtuální stroje. Tento program rozděluje zdroje mezi všechny aktivní virtuální stroje. Všechny operace, které virtuální stroj na *node* provádí, jsou řešeny přes NM. NM poskytuje rozhraní mezi jednotlivými virtuálními stroji a *node*, na kterém tyto stroje běží. Proces NM musí vždy běžet v *root* režimu. *Node manager* zajišťuje veškerou komunikaci a ovládání fyzického stroje, zatímco virtuální stroj slouží jako rozhraní, se kterým pracuje uživatel systému.

4.3.4 Slice

Slice je množina virtuálních strojů, kde každý tento virtuální stroj běží na jiném *node*. V systému je mnoho *slices*. Jeden *slice* = jeden uživatelský projekt. Jednotlivé virtuální stroje, které tvoří *slice* neobsahují žádné informace o jiných virtuálních strojích z této množiny, kromě služby, která zajišťuje běh *slice*. Každý virtuální stroj je spuštěn s množinou klíčů, které umožňují uživateli vzdálené přihlašování a s uživatelsky definovaným *boot* skriptem. Jinak je veškeré další nastavení zajišťováno přes *slice*. To bude popsáno dále v odstavci 4.3.7, věnovaném *slice autority*.

Každý *slice* je jednoznačně identifikován jménem. V tomto jménu je obsažena hierarchie odkazující na vlastníka tohoto *slice*. Např. `cesnet_s1`. Vlastníkem je Cesnet a přesné jméno

slice je s1. Uživatelé přiřazení do jednotlivých *slice* jsou přímo zodpovědní nadřazené autoritě. V tomto příkladu CESNETU. Délka jména jednotlivých *slices,* je však omezena z důvodu přihlašování na 32 znaků.

4.3.5 Služba tvorby slice (Slice Creation Service)

Jedná se o infrastrukturní službu, běžící na každém *node*. Tato služba je spouštěna z PLC a je zodpovědná za tvorbu lokálních instancí *slices* na virtuálních strojích. Tato tvorba probíhá tak, že služba, zavolá lokální manažer uzlu a ten vytvoří nový virtuální stroj. Zde jsou nainstalovány komunikační klíče a je spuštěn odpovídající *boot* skript pro daný *slice* a daného uživatele, čímž dává uživatelům obsaženým ve *slice* přístup k tomuto virtuálnímu stroji.

4.3.6 Služba auditu (Auditing service)

Na každém *node*, běží služba auditu. Ta slouží, k monitorování provozu tohoto *node*. Tato služba shromažďuje informace o komunikaci probíhající na daném *node*. Shromažďují se informace o přenesených paketech z *node*. Služba auditu je dále zodpovědná za monitorování veškeré síťové aktivity, kterou generuje *slice*. Služba auditu také dohlíží na to, aby veškerá aktivita v systému byla mapována na jednotlivé uživatele. Každý paket náleží do určitého *slice* a tento *slice* v danou dobu obsluhoval přesně identifikovaný přihlášený uživatel. Každý paket musí obsahovat zdrojovou adresu a dobu.

Auditorská služba poskytuje webové rozhraní, pro každý *node*, kde se dá najít který uživatel je zodpovědný, za který nežádoucí síťový provoz. PlanetLab exportuje tyto SQL tabulky a ty jsou periodicky kontrolovány autoritou PLC.

Logují se všechny přenesené pakety, ale z důvodů úspory místa jsou tyto informace po určité době mazány. Jedná se o jeden měsíc. Pokud však PLC najde při kontrole provozu nějaké nesrovnalosti, udržují se záznamy déle.

4.3.7 Slice Authority

Slice Autority je orgán dohlížející na správu veškerých *slices* v systému. Tuto roli na sebe bere PlanetLab concorsium a zajišťuje tak správu všech *slices* v systému. Architektura systému však umožňuje, aby zde bylo více *Slice authorities*, ale v praxi vystupuje pouze PlanetLab conconsium. *Slice Authority* také obsahuje a spravuje databázi, která shromažďuje trvalé informace o všech *slices*.

Příklad záznamu v databázi *slices.*

```
principal = (name, email, org, addr, key, role)
org = (name, addr, admin)
slice = (state, rspec)
role = (admin/user)
state = ((delegated/central) & (start/stop) & (active/deleted))
```

Slice autority také poskytuje rozhraní, pomocí kterého uživatelé zadávají uživatelské informace, vytvářejí nové *slice*, přiřazují do *slices* jednotlivé *nodes* a přiřazují k nim uživatele.

Současná implementace zahrnuje dvouvrstvý validační proces pro role administrátorů *slices*. Nejprve PlanetLab concorsium uzavře dohodu s externí organizací, která bude připojena do systému PlanetLab. V této dohodě jsou uvedeny všechny osoby, které jsou zodpovědné za přidávání dalších uživatelů z této organizace. *Slice Autority* potom přidělí těmto uživatelům administrátorská práva k *slices* dané organizace. Tito uživatelé potom mohou na *slices* dané organizace provádět veškeré administrátorské zásahy. Například definovat *RSpec*, přidělovat uživatele nebo určovat, které node budou patřit do kterých *slices*.

RSpec, které spravuje *Slice autority*, je vlastně množina všech *RSpec* definovaných pro použití v *node manager*. To dovoluje *Slice Autority* používat stejnou strukturu pro zaznamenávání doplňujících informací o *slice* a ty potom předat dál službě pro tvorbu *slices*, která běží na každém n*ode*. Například `RSpec` uložené v databázi *Slice Authority* obsahuje jméno, množinu přiřazených uživatelů a množinu *nodes* nad kterou *slice* běží. Dále také může obsahovat atributy, pomocí kterých se dá zjistit, kdy byl *slice* smazán nebo obnoven po předchozím smazání. Některé atributy v uložené v databázi mohou být měněny administrátorem, jiné pouze operátorem PlanetLab concorsia.

Dále pro každé *slice* spravuje *Slice autority* pole *State*, které obsahuje několik klíčových vlastností každého *slice*. Pole *State* je kombinací několika nezávislých hodnot, které popisují důležité vlastnosti *slice.* První bit zde indikuje, jestli tvorba virtuálního stroje pro daný *slice* byla delegována na jinou službu nebo musí být zajištěna pomocí služby *slice autority* spouštěné na každém *node.* Druhý bit indikuje, jestli je *slice* v dané chvíli spuštěn nebo zastaven. Třetí bit indikuje, jestli je *slice* aktivní nebo byl smazán. Pro potřeby auditní služby jsou v databázi udržovány záznamy i o smazaných *slices*.

Existují dvě možnosti spuštění *slice* nad množinou *nodes.* Buďto přímo nebo pomocí delegace. Pro každou možnost však napřed musí být *slice* zaveden do databáze *slice autority* a musí být inicializován *RSpec*. To zahrnuje následující volání v *Slice autority*.

Ukázka kódu pro zavedení *slice* do databáze.

```
CreateSlice(auth, slice name)
SetSliceAttr(auth, slice name, attribute)
AddSlicePrincipal(auth, slice name, principals[ ])
AddSliceNode(auth, slice name, nodes[ ])
```

První řádek zavede *slice* do databáze *Slice autority*, vícenásobné volání druhého řádku vyplňuje *RSpec* pro *slice* a třetí a čtvrtý řádek přiřazují množinu uživatelů a *nodes* nad kterými bude *slice* běžet. Atribut *auth* použitý na všech řádcích obsahuje autorizační údaje daného volání.
Jakmile je *slice* kompletně zaveden do databáze, může být spuštěn pomocí dvou rozdílných volání:

1. Možnost automatického spuštění pomocí příkazu:

```
StartSlice(auth, slice name)
```

Tato operace, nastavuje atribut *state* na *central*, což znamená, že tento *slice* bude obsažen v množině *slices,* které budou vytvořeny na odpovídajících *nodes*. To zajišťuje služba *slice*

creation service, která běží na všech *nodes,* které náleží pod správu dané S*lice Authority*. Tato služba periodicky kontaktuje *Slice Authority* a zjišťuje, jestli existují v množině nějaké *slices*, které by měly běžet na daném *node*, ale zatím tam ještě nejsou zavedeny.

2. V druhém případě uživatel dostává od *slice autority* přidělen *ticket*, pomocí příkazu

```
ticket = GetSliceTicket(auth, slice name)
```

Potom je uživatel zodpovědný sám za kontaktování jednotlivých *slice creation services* běžících na jednotlivých *nodes*, které by měly patřit do daného *slice*. To může být zajištěno i pomocí externí služby dodávané třetí stranou. *Ticket* obsahuje *RSpec* pro daný *slice*, který byl podepsán *Slice Autority* za použití privátního klíče, který odpovídá veřejnému klíči obsaženému v SSL certifikátu *Slice autority* serveru.

Nakonec každá S*lice Authority* komunikuje pomocí SSL certifikátu s vlastníky *nodes*, na kterých *slices* běží. Každý vlastník node, obsahuje tento certifikát v `RSpec`. Služba *slice creation service* také ke komunikaci využívá tento certifikát.

Díky podpoře obou operací, `GetSliceTicket` a `StartSlice` umožňuje systém PlanetLab rozvoj do dvou rozlišných směrů. V jednom *Slice Authorities* odesílají *tickets* a už se nestarají o tvorbu *slices* a v druhém naopak zajišťují kompletní postup tvorby *slices*. V současné době je většina *slices* vytvářena přímo PlanetLab concorsiem. Ale začínají se objevovat pokusy i o externí tvorbu *slices* např. pomocí systému Emulab. Tento postup je popsán v kapitole věnované systému Emulab.

4.3.8 Management Authority

PlanetLab consorsium vystupuje také v roli *Management Authority*. Spravuje server, který instaluje a spravuje softwarové vybavení běžící na *nodes* jím spravovaným. Dále monitoruje chování těchto *nodes* a zajišťuje příslušné akce, pokud odhalí nějaké anomálie nebo chyby. Tak jako pojem *Slice Autority* je pojem *Management Authority* používán jak pro server zajišťující činnost, tak pro organizaci, která jej spravuje. *Management Authority* spravuje databázi registrovaných *nodes*. Každý *node* má přidruženou svou domovskou organizaci (vlastníka), kde je také umístěn. Současná verze databáze obsahuje následující atributy:

```
principal = (name, email, org, addr, keys, role)
org = (name, address, admin, sites[ ])
site = (name, tech, subnets, lat long, nodes[ ])
node = (ipaddr, state, nodekey, nodeid)
```

Kde:

```
state = (install/ boot / debug)
role = (admin/ tech)
```

Pole *admin* obsahuje tak jako u databáze *Slice Authority* odkaz na hlavního administrátora v dané organizaci. Další možnou rolí je role *tech*. Jedná se o technika v dané organizaci. Ten může definovat jednotlivým *nodes* organizace specifickou konfiguraci, která je potom využívána službou *slice creation service* při spouštění node.

Pole *State*, signalizuje, jestli je při dalším restartu *node* potřebná i reinstalace nebo *debug* mode, ve kterém je možno kontrolovat a opravovat *node* bez toho, aby se na něm vytvářely nějaké *slices* nebo se generoval nějaký síťový provoz.

V současné době se využívá jedné databáze jak pro *Slice Authority,* tak pro *Management Authority,* protože obě funkce zajišťuje PlanetLab consorsium.

Management Authority podporuje dvě různá rozhraní pro komunikaci. Buď Public nebo Boot Management.

4.3.8.1 Public interface

Toto rozhraní využívají vlastníci *nodes* k registraci jejich *nodes* u PlanetLab concorsia. Jedná se o offline proces, kdy PLC komunikuje s vlastníky *nodes* a zjišťuje jednotlivé zodpovědné osoby a kontaktní informace o připojovaných *nodes*. Tyto zodpovědné osoby potom využívají tento interface k nahrávání veřejných klíčů do *Management Authority* databáze a vytvoření nových záznamů o jimi připojovaných *nodes*. Ukázka příkladu nahrání klíčů do databáze.

```
node id = AddNode(auth, node values)
UpdateNode(auth, node id, node values)
DeleteNode(auth, node id)
```

Kde `node_id` je jednoznačný identifikátor node, který se dále skládá ze jména a IP adres.

Tento veřejný interface dále podporuje operace, které umožňují uživatelům nebo S*lice Authorities* zobrazovat seznam jimi spravovaných *nodes* a skupin *slices*, které nad nimi běží.

```
node ids[ ] = GetManagedNode(auth)
node values[ ] = GetNodes(auth, node id)
```

Ukázka příkazů volaných na Public interface. Tyto příkazy vrací DNS jméno a IP adresy specifikovaných *nodes*.

4.3.8.2 Boot Manager Interface

Nodes využívají druhé rozhraní *Management Authority* ke kontaktování PLC při jejich startování. *BootFile,* obsažené na každém *node* obsahuje minimální Linux, který inicializuje HW vybavení *node,* načte síťovou konfiguraci ze souboru *plnode.txt* a kontaktuje PLC. V této chvíli Management autority vrací spustitelný program nazývaný *boot manager* (asi 20Kb kódu), který *node* ihned spouští.

Boot manager, běžící na *node* přečte *nodekey* ze souboru *plnode.txt* a použije HMAC pro autentizaci proti Management autority s tímto *nodekey*. Tento způsob komunikace se využívá i při další komunikaci. MA – *Management Authority* dále ověřuje, jestli je současná IP adresa *node* stejná jako ta uložená v databázi, kvůli ověření, jestli byl správný soubor *plnode.txt* použit pro odpovídající *node*.

První věc, kterou si *node,* od MA načítá, je aktuální *state*. Jestli se *state* = *install* tak se spouští instalační program, který formátuje disk a stahuje si aktuální VMM – Správce virtuálních obrazů, NM - *node manager* a další potřebné balíčky od *Management Authority*. Jakmile jsou balíčky staženy a nainstalovány, spouští se kernel a mění se *state* v databázi MA na *boot*.

Pokud je node ve stavu *boot*, tak *boot manager* generuje privátní a soukromé klíče používané při autentizaci *slices* . Jakmile jsou klíče vygenerovány, musí se spárovat s databází MA. Potom se kontaktuje MA s požadavkem na aktualizaci packages. Pokud aktualizace není potřebná, začíná se se zaváděním odpovídajících *slices*, které tento *node* používají.

Pokud je *node* ve stavu *debug*, pokračuje *boot manager* se zaváděním kernelu, ale povolí se pouze inspekční zásahy operátorů do *node*. Tito operátoři také mohou nastavovat jednotlivé stavy v MA databázi.

Mimo startování existují ještě další dva stavy, kdy se node synchronizuje s MA.

1. *Node,* periodicky kontaktuje MA, jestli nepotřebuje stáhnout nové updaty softwaru nebo jestli nepotřebuje reinstalaci. Toto probíhá u každého *node* 1x denně.
2. Pokud je stav node v MA databázi nastaven na *debug*, MA kontaktuje *node* a ten spouští *boot* proces, aby byl nastaven do tohoto stavu co nejrychleji.

4.3.9 Vlastník zdrojů (Owner)

Architektura systému PlanetLab poskytuje vlastníkům zdrojů co nejvíce samostatnosti při správě svých *nodes*, aby bylo dosaženo co největšího snížení nákladů na správu na straně PLC. Proto vlastníci zdrojů, potřebují nějaký způsob pro komunikaci s *node* při jeho správě. Např. vlastník může chtít předepsat, které *slice* se budou moci na jeho *nodes* spouštět a přidělit zdroje mezi jednoho nebo i více *Slice Authorities,* pro následnou redistribuci mezi *slices*. Tyto nastavení, pro každý *node,* zajišťují *site_Admins,* popsaní v předcházející kapitole. Jejich požadavky se zpracovávají pomocí *owner script* [5]běžícím na Virtuálním stroji VM. Tyto skripty v současné době vytváří a vzdáleně potom na příslušných *node* spouští PLC.

Vlastník zdrojů obyčejně přiděluje skupinu zdrojů, buď Slice Authorities nebo samostatným *slice*. To zajišťuje operace `CreatePool()`. Tato operace obsahuje argument *slicename,* pomocí kterého se určuje, kterému *slice* je povoleno načítat `rcap` a `RSpec` pomocí metod `GetRcap()` a `GetRspec()`. V *RSpec* jsou v této chvíli obsaženy tři informace.

[5] Každý vlastník fyzického zdroje si definuje vlastní politiky, které se budou na jeho hw uplatňovat. Owner skript slouží pro jejich popis.

1. Specifikace zdrojů obsažených v *Pool.*

2. Jméno *Slice Authority*, která je oprávněna vytvářet *slice* za použití tohoto *Pool*.

3. SSL certifikát určený pro komunikaci s *Slice Authority*.

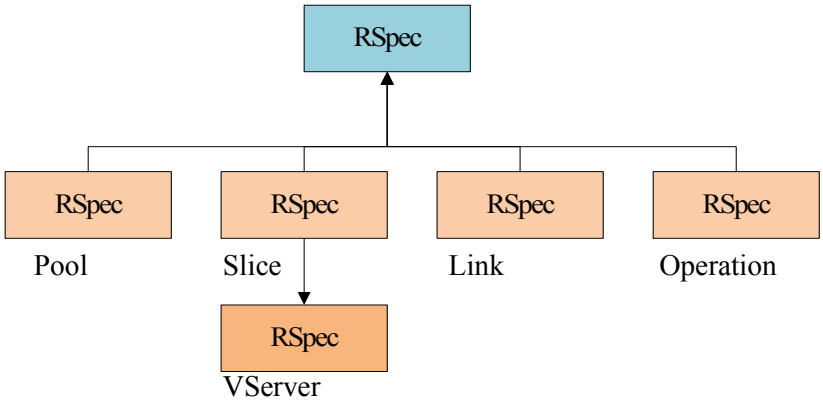
4.3.10 Specifikace zdrojů, RSpec

Resource Specification – Specifikace zdrojů (*RSpec*), je abstraktní objekt používaný v systému PlanetLab. Jedná se o hlavní objekt, spravovaný *Node Managerem NM* a *Slice creation service*, který se využívá jako primární komponenta záznamu v databázi *Slice Authority* pro každý *slice*.

Na *RSpec* se dá také nahlížet, jako na sadu atributů nebo skupiny jmen a hodnot. Například následující skupina atributů odpovídá defaultnímu nastavení virtuálního stroje VM.

```
cpu share = 1
disk quota = 5 (GB)
mem limit = 256 (MB)
base rate = 1 (Kbps)
burst rate = 100 (Mbps)
sustained rate = 1.5 (Mbps)
```

Navzdory tomuto příkladu, může mít *RSpec* i určitou strukturu. Ve většině případů se na něj dá pohlížet jako na třídu v objektové hierarchii jako na obrázku 5.



Obrázek 5: Struktura RSpec

PoolRSpec – seznam aktuálních zdrojů.

SliceRSpec – umístění *slice* ve struktuře PlanetLab.

VServerRSpec – RSpec virtuálního serveru obsluhujícího jednotlivé *nodes* ve *slice.*

LinkRSpec – odkazy na ostatní *RSpec*, v případě nutnosti přerozdělení zátěže.

OperationRSpec – funkční objekt který povoluje klientovi provádět privilegované operace.

Implementace systému podporuje do této hierarchie ještě doplňovat další třídy v případě potřeby.

Protože se souboru *RSpec* využívá pro komunikaci mezi různými objekty *Slice Auhtority* a *Slice creation service* je nutné definovat přesný způsob a formát přenosu informací. *RSpec* je v současné době reprezentováno XML souborem.

4.3.11 Bezpečnostní architektura

Tato podkapitola obsahuje bezpečnostní architekturu jednotlivých komponent systému popsaných níže. Je zde popsáno, jak *nodes* zabezpečeně startují, jak se vytvářejí a zpřístupňují *slice* a jak se mezi sebou autentizují jednotlivé služby v rámci celého systému.

4.3.11.1 Statické klíče

Malé množství veřejných klíčů a certifikátů musí být vytvořeno a redistribuováno mezi různé komponenty infrastruktury k zajištění základního autentizačního rámce. Následující požadavky musejí být splněny:

Musí existovat známý, přístupný seznam veřejných klíčů, které slouží jako *root keys,* které se používají k identifikaci *Top-level* certifikátů. Tato množina klíčů obsahuje klíče vydané komerční certifikační autoritou. (VeriSign)

Každá *management* a *slice autority,* která poskytuje rozhraní pro přístup pomocí služeb (např. běh *management service* na vlastních *slices*, musí mít vygenerovaný pár public – private klíčů a certifikát podepsaný jednou z certifikačních autorit CA, která může být autentizována pomocí odpovídajících veřejně známých klíčů.

4.3.11.2 Certifikáty

Jako formát certifikátů se používá XML dokument odpovídající XML-SEC standardu. Každý takový dokument se skládá ze tří částí:

Subject: Identifikuje subjekt a obsahuje PEM reprezentaci veřejného klíče subjektu. Certifikát předepisuje, jak se ověřuje spojení mezi identitou subjektu a veřejným klíčem verifikovaným pomocí operačního systému.

Issuer: Data využívaná k autentizaci certifikátu, buď jako identifikátor dobře známého veřejného klíče – to znamená, že se jedná o *top level* certifikát používaný jako *root key* nebo certifikát, který obsahuje veřejný klíč autority, která jej podepsala.

Signature: Specifikuje počet podpisových parametrů (*signature parameters*) a obsahuje kryptografický podpis hodnoty tohoto certifikátu. Autorita využívá tyto informace při potvrzování pravosti tohoto certifikátu.

V současné době se v systému využívá jednoduchý Python skript `mkcert` pro generování a potvrzování XML certifikátů. To dovoluje skrýt obsah certifikátů před koncovými uživateli.

Aktuální podepisování a verifikace jsou postaveny na použití utility `xmlsec1`, která využívá OpenSSL. V budoucnu se uvažuje o rozšíření podpory certifikátů o standart x509.

4.3.12 Bootování node

Každá *Management Authority* provozuje *boot server,* který kontaktují *nodes* klientů, když provádějí reboot. Každý node bootuje z neměnného souborového systému, typicky CD-ROM, pomocí kterého se zavádí *boot manager* na *node*. *Boot manager* potom kontaktuje *boot server* spuštěný na *management authority*. Pokud je to třeba, stahuje si potřebný kód a konfigurační informace potřebné ke spuštění node. Neměnný souborový systém dále obsahuje množinu certifikátů potřebných pro autentizaci proti *boot serveru*. Informace uložené na bootovacím CD jsou specifické pro každou *Management Authority*, ne pro každý *node*.

Jakmile je node registrován u *Management Authority*, jsou generovány privátní klíče pro *node*. Ty jsou potom exportovány jako část *node configuration file*. *Node key* je následně použit boot managerem, jako *shared secret* k autentizaci *node* u *Management Authority*. Vlastník *node* je povinen zkopírovat konfigurační data na médium, které bude následně chráněné proti zápisu. Dále se předpokládá, že *node* je na bezpečném místě a tudíž k němu nemají přístup neautorizované osoby.

Jakmile *node* nabootoval, je třeba znova zabezpečeně kontaktovat *Management Authority* pro stažení potřebného kódu a aktuálních konfiguračních informací. V tomto případě *node,* využívá sadu CA certifikátů k autentizaci proti *boot serveru Management Authority* a dále používá jeho tajný *node key,* pro autentizaci proti *Management Authority* s použitím HMAC. Autentizovaný *node,* následně upozorní *Management Autority*, že je správně nabootovaný a požádá o *standard* nebo *debug mode* (podle aktuálního nastavení).

4.3.13 Tvorba slice

Tvorba *slice* je více stavový proces zahrnující spolupráci vlastníků *node*, *slice creation sevice* a *Slice Authority*. Proces sestává z následujících kroků:

Vlastník *node* požaduje SSL certifikát pro S*lice Authority*. Pro každou S*lice Authority* vlastník vytváří konfigurační soubor, který identifikuje *Slice Authority* a obsahuje SSL certifikát pro tuto autoritu. *Node manager* při každém bootování spustí *owner skript,* který obsahuje údaje o `RSpec` a identifikuje službu *slice creation service*. Tuto službu mohou spouštět pouze virtuální stroje VM patřící danému vlastníkovi.

Jakmile je služba *slice creation service* nastartována, zavádí se množina *slice pools* vytvořených vlastníkem *node* a spojených s danou službou. Dále se vytváří pár privátních/veřejných klíčů. Pro každý *pool* se využívá nezávislého páru klíčů pro autentizaci odpovídající *Slice Autority,* která spravuje sady *slices* pro všechny jím spravované *node*. Jakmile je S*lice Authority* autorizována, probíhá vytváření virtuálních strojů VM. Vždy jeden VM na jeden *node* pro každý *slice*.

Slice Authority dále používá soukromý klíč odpovídající veřejnému klíči v jejím certifikátu pro ověřování `tickets`, pomocí kterých se následně přidělují odpovídající fyzické zdroje

patřičným VM, respektive *slices*. Tyto `tickets` jsou generovány a následně podepisovány přímo uživateli *slices*.

4.3.14 Povolování uživatele

PLC autorizuje *sites* a jejich PI pomocí offline autorizačního procesu. Na základě této autorizace je potom vytvořen účet PI pro odpovídající *slice* a *Management Authorities* daného vlastníka. PI následně využívá zabezpečené spojení s PLC pro nahrání veřejného SSH klíče, který je následně nainstalován na všechny VM daného vlastníka.

Uživatelé při tvorbě svých uživatelských účtů využívají stejný princip, ale ještě je musí následně schválit PI. Jakmile dojde schválení a povolení uživatelského účtu, musí se také nahrát SSH klíč, který je následně redistribuován na všechny VM které patří do *slices* daného uživatele. Tento klíč se potom využívá pro zabezpečenou komunikaci se *slice*.

4.3.15 Autentizační a autorizační metody založené na certifikátech

V systému se využívá také možnosti autentizace a autorizace pomocí certifikátů. Systém nemůže požadovat po všech používaných službách, aby si generovaly sady privátních a veřejných klíčů pro každý *node*, se kterým budou komunikovat. Z tohoto důvodu se zavedly certifikáty.

Služba napřed vytvoří pár klíčů, soukromý a veřejný, poté požaduje po *node manager,* aby podepsal certifikát spojující veřejný klíč se službou. *Node manager* využívá nízko úrovňové, VNM - *Virtual node* rozhraní pro identifikaci původu dotazu *slice*. *Node manager* potom generuje certifikát, spojující veřejný klíč se *slice* a podepíše jej pomocí jeho veřejného klíče.

Pokud služba požaduje zavolání nějaké operace na *Slice* nebo *Management Authority*, musí se nejprve autentizovat a získat *session key*, pomocí kterého se bude potom komunikovat. Tato autentizace se provádí zavoláním příkazu:

```
session key = AuthenticateCertificate(cert)
```

Kde se certifikát použije jako parametr. Autorita následně verifikuje certifikát a vytvoří *session key,* který odpovídá žadateli. Tento *sesion key* se následně zašifruje pomocí veřejného klíče obsaženého v certifikátu a odešle zpět žadateli. Ten si jej potom rozšifruje pomocí svého privátního klíče. Tím je zaručena bezpečná výměna informací.

4.3.16 Rozhraní

Tato kapitola popisuje vnější rozhraní systému PlanetLab. Tato rozhraní slouží pro komunikaci s uživateli nebo externími systémy.

Node manager - Poskytuje rozhraní používané infrastrukturou služeb pro vytváření a manipulaci s virtuálními stroji VM a *ressource pools*. Toto rozhraní může být voláno pouze lokálně. Většina operací se provádí pomocí specifikace `RSpec` jako argumenty.

Slice creation service – Poskytuje vzdáleně přístupné rozhraní volané *Slice* autoritami a uživateli pro tvorbu nových *slices*.

Slice authority – Poskytuje rozhraní pro registraci nových uživatelů, tvorbu nových *slices* těmito uživateli. Dále rozhraní pro *Management Authority* pro získávání informací o množině uživatelů přiřazených k jednotlivým *slices*.

Management authority – poskytuje dvě rozhraní. Veřejné a soukromé. Veřejné rozhraní je využíváno pro registraci nových *nodes*. A dále jej využívají uživatelé a S*lice Authorities* pro zjišťování množiny *nodes* spravovaných danou autoritou. Soukromé rozhraní využívají *nodes* pro stahování a instalaci softwaru a konfiguračních dat.

PlanetFlow Archiving service - poskytuje rozhraní odpovídající každému *node*, z kterého se dají zjistit auditní záznamy o komunikaci.

4.3.17 Organizační principy

Systém Planetlab je založen na pěti základních organizačních principech vycházejících z požadavků uživatele a možností samotného systému. V této kapitole budou tyto principy popsány a bude zde nastíněno řešení, jak systém tyto principy implementuje a používá.

4.3.17.1 Distribuovaná virtualizace

Hlavním cílem systému PlanetLab je poskytování stabilní platformy, která poskytuje široké spektrum služeb, které jsou založeny na připojování pomocí vzdálených, vícenásobných a nezávislých přístupových bodů připojených do celosvětové sítě. Tento systém je postaven na tzv. uživatelském módu. V tomto módu pracují většinou výzkumníci a poskytovatelé služeb. Tento mód je základem pro krátkodobé experimenty, řádově několik týdnů nebo měsíců. Každý projekt běží na nějakém *slice*, který je součástí globálního systému. V tomto globálním systému může vedle sebe běžet mnoho takovýchto projektů nezávisle na sobě. Každý samostatný *slice* sdružuje několik uživatelů a několik *nodes*. Jeden uživatel může být součástí i více *slices* a taktéž jeden *node* může být obsažen ve více *slices*. Distribuovaná vizualizace je zde založena na tom, že jednotlivé *slice* zde běží nezávisle na sobě a izolovaně od druhých a zároveň mohou být sdíleny více uživateli a může k nim být přistupováno odkudkoli.

4.3.17.2 Oddělené řízení

PlanetLab čelí dilema. Je navržen pro poskytování platformy pro výzkum v celosvětovém měřítku velkému počtu nezávislých uživatelů, ale také jeho systém řízení je distribuován mezi široké spektrum nezávislých uživatelů. Z tohoto důvodu bylo v minulosti nutné nasadit PlanetLab a začít získávat zkušenost s poskytováním různých síťových služeb, než byly odhaleny principy potřebné pro dokonalé řízení takto rozsáhlé platformy. V důsledku toho musel být systém Planetlab navržen s podporou pro následnou evoluci těchto řídicích systémů. Kvůli tomu je systém řízení rozdělen do kolekce většinou nezávislých služeb, z nichž každá běží nezávisle na jiné. Většina z těchto služeb byla vyvinuta třetí stranou a je pouze nasazena a využívána systémem. Mezi hlavní služby řízení patří například tvorba *slice* nad skupinou *nodes*, management připojování nových

nodes do systému, udržování a správa kódu na jednotlivých *nodes*, monitorování chování *slice* z hlediska bezpečnosti atd.

4.3.17.3 Řetěz zodpovědnosti

Systém PlanetLab využívá mnoha *nodes,* které jsou poskytovány výzkumnými organizacemi po celém světě. Na těchto *nodes* běží různé služby spuštěné uživateli z jiných výzkumných organizací. Jednotliví uživatelé nemusejí přímo znát vlastníky jednotlivých *nodes* v systému. Aby byla situace ještě komplikovanější, je možné pomocí spuštěných služeb odeslat potencionálně nebezpečné pakety do internetu. PlanetLab Concorsium zde hraje roli důvěrného zprostředkovatele, tudíž jednotliví poskytovatelé *nodes* nemusejí uzavírat dohody s jejich uživateli. Důležitou povinností systému PlanetLab je udržovat řetězec zodpovědnosti mezi všemi poskytovateli a uživateli. To znamená, že musí být možné mapovat externě viditelné činnosti např. posílání paketů a musí se dát vysledovat, který uživatel je za který paket zodpovědný. Tato schopnost má zásadní význam pro zachování důvěryhodnosti vztahů mezi jednotlivými stranami. Tato funkcionalita nevylučuje, že se nemohou vyskytnout nějaké problémy, ale poskytuje možnost vystopovat uživatele, který je za tyto problémy zodpovědný.

4.3.17.4 Decentralizované řízení

PlanetLab je celosvětovou platformou složenou z komponent, které jsou ve vlastnictví mnoha autonomních organizací. Každá organizace si musí zachovat určité množství kontrolních mechanizmů nad tím, jak jsou její prostředky používány a zároveň PlanetLab jak celek musí poskytovat určitou možnost vzdálené správy jednotlivých zařízení a celého systému. V důsledku toho musí PlanetLab podporovat decentralizované řízení, což vyžaduje, aby pouze minimum akcí potřebovalo globální schválení. Systém to dělá tak, že umožňuje sdružování autonomních organizací do větších globálních celků a mezi těmito celky jsou potom definovány jednotlivé vztahy.

4.3.17.5 Efektivní sdílení zdrojů

Protože PlanetLab pracuje jako dotované prostředí, musí se pořád potýkat s nedostatkem prostředků. Proto je nezbytné silně podporovat sdílení zdrojů. V současné době systém pracuje na dvou základních systémech sdílení zdrojů.

1. Striktní oddělení vytváření nových *slice* a následné přidělování fyzických zdrojů k tomuto *slice*. To znamená, že všechny *slice* při svém vytvoření použijí všechny požadované fyzické zdroje. Jakmile však nejsou potřeba všechny fyzické zdroje, ty nepotřebné jsou uvolněny a distribuovány dále pro jiné *slice* pomocí "makléřské služby - *brokerage service*". Neplatí tedy, že *slice* je pevně svázán se všemi fyzickými zdroji po celou dobu své existence.
2. I když některé *slice* mohou získat „ zaručené zdroje" na celou dobu jejich existence, očekává se, že i tyto zdroje budou využívány pouze po dobu nezbytně nutnou.

V praxi to funguje tak, že 99% *slices* je vytvářeno v standardním režimu, viz 1. Při jejich vytvoření se na chvíli alokují všechny požadované zdroje. Následně je zjištěno, že některé fyzické

zdroje nejsou potřeba, tak jsou dynamicky uvolňovány. V druhém případě, při využití zaručených zdrojů se dbá na to, aby tyto zdroje byly využívány efektivně, tudíž, aby se alokovaly pouze na nezbytně nutnou dobu.

Planetlab taktéž obsahuje mechanizmy, pomocí kterých je schopen provést automatické zotavení po přetížení některého nebo i více fyzických zdrojů. Pokud je některý *node* po delší dobu vytížen nějakým procesem na 100 % a není k tomu důvod, není zde žádný *slice* s přihlášeným uživatelem, je tento proces automaticky odpojen.

4.4 Práce se systémem

Tato kapitola se bude věnovat připojení do systému PlanetLab, založení nového účtu a základnímu popisu přístupu do systému. Dále bude uveden jednoduchý příklad použití systému. Veškeré zde uvedené informace vycházejí z obecných návodů pro systém a jsou prakticky odzkoušeny.

4.4.1 Registrace nového účtu

Pro přihlášení do systému Planetlab je nejprve nutné provést registraci na stránce http://planet-lab.org. Na hlavní stránce, v pravé horní části je záložka *create new account*. Po kliknutí na tuto záložku stačí vyplnit formulář a vybrat, pod kterou instituci spolupracující s Planetlabem patříte. Je potřeba vyplnit všechny položky formuláře a zadat emailovou adresu, která bude sloužit pro komunikaci a zároveň jako přihlašovací jméno pro přístup do systému. V mém případě, jsem jako hostující instituci, *site* vybral CESNET. Někdo z hostující instituce musí registraci napřed potvrdit. S potvrzením pro akademickou výzkumnou činnost není problém. O následné registraci budete informováni pomocí emailové zprávy.

V mém případě proběhla registrace rychle a bez problémů. Do druhého dne po odeslání žádosti, jsem měl přidělený účet, schválený jak od hostující instituce, tak od systému PlanetLab. Jako hostující instituci jsem zvolil CESNET.

Jakmile proběhne potvrzení hostující institucí, můžete se do systému přihlašovat pomocí zadaného emailu a hesla.

4.4.2 Přihlášení

Pro přihlášení je třeba vyplnit emailovou adresu a zvolené heslo na hlavní stránce Planetlab. Po úspěšném přihlášení budete přesměrováni na hlavní stránku webového rozhraní Planetlab. Ta v pravé horní části obsahuje několik nabídek, ve kterých jsou uvedeny důležité informace a nastavení. To bude popsáno dále v textu.

4.4.3 Vložení SSH Klíče

Než se systémem Planetlab začnete pracovat, budete potřebovat svůj SSH veřejný klíč. Tento klíč se používá pro přístup k jednotlivým n*odes* v síti Planetlab. Pokud nemáte vygenerován vlastní SSH klíč můžete použít příkaz:

```
$ ssh-keygen -t rsa -f ~/.ssh/id_rsa
```

Jakmile máte vytvořen pár SSH klíčů, je třeba nahrát veřejný klíč do databáze systému Planetlab. Pro nahrávání se musíte nejprve přihlásit pomocí tlačítka *login* a vašich přihlašovacích údajů. V levé části stránky bude zobrazen váš účet. Zde vyberte záložku *My Account*. V sekci *Keys* vyberte *Manage Keys*, Klikněte *Browse* a vyberte umístění vašeho veřejného SSH Klíče. Nakonec knikněte na *upload* a zkontrolujte, že byl klíč úspěšně nahrán. Musí se také dát pozor na to, aby nahraný klíč odpovídal klíči počítače, pomocí kterého budete potom se systémem komunikovat. Může se totiž stát, že přihlašování do webového rozhraní a samotná komunikace pomocí příkazové řádky probíhá na rozdílných počítačích. To by potom přihlášení a ověření nefungovalo.

4.4.4 Získání Slice

Všechny přístupy do sítě Planetlab jsou řešeny přes *slice*. *Slice* je kolekce zdrojů distribuovaná mezi více *nodes* Planetlabu. Když se *node* přidá do *slice* , vytvoří se virtuální server pro tento s*lice*. Když je *node* odebrán, virtuální server se zruší. Všechny *slice* které používáte, spravuje váš PI. V mém případě jsem byl přiřazen do testovacího slice `cesnet_s1`. Jakmile jste přiřazen do *slice*, můžete si na stránce my *Account* v záložce *slices* prohlížet všechny *slice,* do kterých jste přiřazeni a upravovat jejich nastavení.

4.4.5 Přidávání Nodes do Slice

Před začátkem práce se systémem Planetlab je třeba si do *slice* přidat nějaké *nodes,* se kterými budeme pracovat. Toto přidání se provádí na webové stránce Planetlabu. Po přihlášení je třeba vybrat menu *slices* a tam vybrat, ke kterému *slice* se chceme připojit.

Tím se dostanete do menu svého *slice*. Zde si můžete prohlížet jednotlivé detailní informace o tomto *slice*. V položce *details* jsou zobrazeny jméno, popis, adresa, datum exspirace a instituce, které *slice* patří. V položce *Users* jsou zobrazeni jednotliví uživatelé, kteří s tímto *slice* pracují. Hlavní položkou, která nás teď bude zajímat, však teď bude položka *nodes*. Pomocí této položky se dají do *slice* přidávat jednotlivé *nodes*. Jsou zde zobrazeny *nodes,* které jsou již přidány a seznam všech použitelných *nodes,* které se dají přidat. U každého *node*, je zobrazen jeho současný stav. Může se jednat o stavy *boot- node* je připraven k použití, *failboot* – nepodařilo se nastartovat a *reinstall* – údržba sw vybavení. Dále je u každého *node* zobrazeno, pod kterou instituci patří, jeho *hostname* a jestli je součástí projektu Planetlab (zkratka PLC) nebo evropského projektu Planetlab Europe (zkratka PLE). Přidání *nodes* se provádí pomocí zaškrtnutí checkboxu u vybraného *node* a následným přidáním pomocí tlačítka *add selected*. Mezi jednotlivými *nodes* se také dá vyhledávat.

Přidávání *nodes* do *slice* musí probíhat v pořadí, v jakém se k nim bude následně chtít přistupovat. Nejjednodušší způsob jak přidat *nodes* je jejich označení a přidání jednoho po druhém na stránkách Planetlab.

Dalším krokem bude vytvoření pracovní složky, pro ukládání pracovních dat na počítači, pomocí kterého budeme k systému přistupovat.. V mém případě se jednalo o složku `~/planetlab/test1/`

4.4.6 Přístup k API pomocí PlanetLab Shell

Nejjednodušší cestou, jak se připojit do sítě PlanetLab je pomocí aplikace *PlanetLab shell*, `plcsh` . Bude potřeba si obstarat instalační soubor z PlanetLab CSV Repository. Nejjednodušším způsobem je použít následující příkazy pro stažení a následnou kompilaci přímo na počítači.

```
$ cvs -d :pserver:anon@cvs.planet-lab.org:/cvs checkout
new_plc_api
$ cd new_plc_api
$ make
```

Jakmile bude aplikace *PlanetLab shell* úspěšně zkompilována, dá se využít pro přidávání a práci s *nodes* v daném *slice*. Aplikace *PlanetLab shell* využívá syntaxi jazyka Python a obsahuje plnohodnotný Python interpreter. Z toho plyne, že jakýkoliv kód napsaný v pythonu se dá pomocí aplikace *PlanetLab shell* spustit.

Příklad pro přihlášení k systému Planetlab a přidání *nodes* ze souboru `nodes.txt` do slice.

```
$ ./plcsh -u //uživatelské jméno -r //uživatel
Password:  //heslo
[]>>> node_list = [line.strip() for line in open("../nodes.txt")]
[]>>> AddSliceToNodes("jmeno_slice", node_list)
[]>>> exit
```

4.4.7 Přístup k API pomocí jazyka Python

Další možností pro přístup k PlanetLabu je přístup pomocí skriptovacího jazyka, který podporuje XMLRPC. Následující příklad bude uveden v jazyku Python a zajišťuje přihlášení k systému a přidání *nodes* do *slice*, tak jako předchozí příklad. Vykonání skriptu trvalo přibližně 20 minut, z důvodu alokace všech virtuálních serverů na všech *nodes* v daném *slice*.

```
$ python
>>> import xmlrpclib
>>> api_server = xmlrpclib.ServerProxy('https://www.planet-
lab.org/PLCAPI/')
>>> auth = {}
>>> auth['Username'] = "přihlašovací jméno"
```

```
>>> auth['AuthString'] = "heslo"
>>> auth['AuthMethod'] = "password"
>>> node_list = [line.strip() for line in open("nodes.txt")]
>>> api_server.AddSliceToNodes(auth,"jmeno slice", node_list)
```

4.4.8 Rozmístění aplikací na Nodes

Existuje více možností jak redistribuovat vlastní software na používané *nodes*. Pro PlanetLab existuje několik nástrojů, které umožní zpracovávat paralelní SSH příkazy nad kolekcí používaných *noodes*. Použití těchto příkazů však není vhodné pro větší počet *nodes*, protože způsobuje nárazově velkou zátěž pro centrální server.

Proto se pro distribuci vlastního sw na *nodes* používá většinou *content distribution network*. Tato technologie vylučuje tvorbu vysoké zátěže. Pro přístup k této technologii se využívá nástroj PlanetLab *CODeploy*.

4.4.9 Instalace CoDeploy

Detailní popis instalace a používání aplikace *CoDeploy* nedostupný na webu http://codeen.cs.princeton.edu/codeploy/, zde bude ukázána pouze základní ukázková konfigurace.

Prvním krokem je stažení *CoDeploy tarball* do pracovního adresáře. Dále je potřeba provést následující příkazy v této složce. Složka by měla taktéž obsahovat soubor, ve kterém jsou uloženy používané *noodes*. `(nodes.txt)`

```
$ tar zxf codeploy.tar.gz
$ cd codeploy
$ make
$ export MQ_NODES='../nodes.txt'  // cesta k souboru nodes.txt
$ export MQ_SLICE='...' //jméno používaného slice
$ echo "StrictHostKeyChecking no" >> ~/.ssh/config
```

Poslední řádek příkazu zastaví pokus o SSH verifikaci každého *node,* který obsahuje *host key*. Bez tohoto příkazu by bylo potřeba při každém přístupu k novému *node* zadávat *host key*. Toto nastavení je sice na úkor bezpečnostního rizika možnosti odposlechu komunikace, ale výrazně urychluje práci se systémem. Z vlastních zkušeností to znamená asi 70 % úspory času.

Aplikace *CoDeploy* je dostupná na webu: http://codeen.cs.princeton.edu/codeploy/

4.4.10 Rozmístění uživatelského skriptu na jednotlivé nodes

Uživatelské skripty se dají rozmístit na *nodes* dvěma způsoby. Buď manuálně, nebo pomocí podpůrného programu.

4.4.10.1 Manuální nasazení vlastního sw na node

Pro přihlašování do požadovaného *node* se používá příkaz:

```
ssh cesnet_eng@planetlab1.cesnet.cz -i ~/.ssh/ident
```

kde –i je parametr, pomocí kterého se zadává umístění vlastního ssh klíče.

4.4.10.2 Rozmístění uživatelského skriptu pomocí CoDeploy

Při použití programu *CoDeploy* je třeba mít přístup k lokální složce, do které by měl být umožněn i přístup přes rozhraní http. Software, který má být nasazen, musí být umístěn v této složce. Dále také program *CoDeploy* bude v této složce ukládat tempomaty files. Ve výchozím nastavení složka vypadá takto:

```
~/public_html/scriptname
http://www.your.url/~username/scriptname.
```

Každý skript se skládá z jednoho nebo více python skriptů, které je potřeba následně rozmístit na všechny *nodes* používané daným *slice*. Nejjednodušší způsob je uložit tyto skripty do sdílené složky popsané výše ve formátu `skript.py`. O rozmístění se dále stará *CoDeploy* pomocí zadaných argumentů.

```
$ ./codeploy ~/public_html/scriptname \
> http://www.your.url/~username/scriptname/ script1
```

4.4.11 Spouštění vlastního software

Ve chvíli, kdy jsou veškeré skripty rozmístěny na *node* je třeba tento sw spustit. To se dělá pomocí utility *MultiQuery* dodávané společně s *CoDeploy*. Jedná se o jednoduchý program, který umožňuje spouštět současně vzdáleně příkazy na *node* pomocí SSH. Program *MultiQuery* se spouští pomocí příkazu:

```
$ ./multiquery 'echo "python hello/ScriptName.py" |
crontab -; sudo /sbin/service crond MethodName'
```

Ve složce, kde je nainstalován *CoDeploy*. Jako argumenty se zde zadává jméno volaného skriptu a metoda, která se má vykonat.

4.5 Závěr

Systém Planetlab, podobě jako systém Emulab je zaměřen spíše na koncové prvky počítačových sítí, tj. koncové počítače. Jeho hlavní prioritou je to, že jednotlivé *nodes* jsou rozprostřeny po celé planetě a díky tomu je tento systém schopen poskytnout reálnou simulaci Internetu. Protože jednotlivé *nodes* využívají pro vzájemnou komunikaci šifrovaného spojení přes internet, neexistuje zde mnoho možností pro nastavování parametrů linek. Testbed je zaměřen hlavně na simulaci chování uživatelského software nainstalovaném na jednotlivých *nodes* v rámci celosvětového internetu. Sofistikovanější síťové prvky jako jsou směrovače nebo přepínače se v tomto testbedu nedají simulovat vůbec. Pro svůj účel, testování chování uživatelského softwaru rozprostřeného v rámci celosvětové sítě, však využívá zajímavé metody, díky kterým může poskytnout dokonale izolované a zabezpečené prostředí, které se však chová jako Internet.

Se systémy vyvíjenými na VŠB se tento testbed nedá srovnávat, protože pracuje na úplně jiném principu a slouží pro naprosto odlišný druh testování síťových řešení.

Díky tomu, že systém PlanetLab je pro akademickou činnost volně přístupný, jedná se velice vhodný nástroj pro testování síťových technologií v reálném prostředí Internetu.

5 Panlab

Systém Panlab není čistý testbed, tak jako jiné systémy popsané v této práci. Panlab v sobě spojuje několik různých testbedů v rámci Evropy a poskytuje tak uživatelům daleko větší spektrum funkcionality a možností, než by mohly jednotlivé tesbedy poskytnout samostatně. Panlab implementuje množství technologií, které zaručují dokonalé propojení těchto izolovaných testbedů a umožňuje tak sestavovat další uživatelské testbedy, které mohou být složeny z mnoha komponent, které jsou rozděleny do několika nezávislých administrativních domén. Systém zajišťuje jejich propojení, konfiguraci a komunikaci s uživatelem. V následujících kapitolách bude popsáno, jak jednotlivé technologie v systému pracují a jak byly implementovány.

5.1 Cíle projektu

Panlab – PLL reaguje na nutnost testování zařízení v oblasti telekomunikací a síťových technologií zavedením a zpřístupněním infrastruktury pro spojování testbedů. V rámci projektu je využito evropského projektu inovačních klastrů a existujících testbedů, které v rámci těchto klastrů již existují. Hlavním cílem systému, je vytvořit federaci testbedů, mezi těmito inovačními klastry, v rámci Evropy. To umožní společnostem, které jsou součástí těchto uskupení, testování nových komunikačních a síťových technologií. Současná federace obsahuje čtyři centrální a tři satelitní clustery.

PLL se snaží vyvíjet a nasazovat technologie pro spolupráci a další rozvoj testbedů v rámci Evropy.

Hlavními cíli systému jsou:

- Rozvoj mechanizmů a nástrojů pro popis, ukládání vyhledávání a organizaci testovacích služeb tak, aby mohly poskytovat jednotné rozhraní nad více různými doménami.
- Definice kontrolního rámce, který bude běžet nad všemi zúčastněnými testbedy a zajišťovat diagnostiku a spolupráci nad různými technologiemi.
- Vytvořit důvěrný vztah mezi organizacemi v rámci federace pomocí procesů a nástrojů pro zvyšování kvality

5.2 Slovník pojmů

Slovník slouží pro základní vysvětlení funkcionality vybraných základních komponent systému. Detailní popis a principy fungování zde zmíněných komponent, bude popsán v kapitole věnované architektuře systému.

Teagle – Hlavní portál systému. Uživatelé se zde registrují, přihlašují a pomocí nástrojů dostupných přes tento webový portál si sestavují vlastní VCT.

VCT – Uživatelský projekt, experiment v systému. Obsahuje *resources* a topologii, na které se pracuje. Pro sestavování vlastních VCT se využívá nástroj VCT tool, dostupný přes portál Teagle.

Resources – Jednotlivá fyzická zařízení obsažená v testbedech připojených do systému Panlab. Vlastníky těchto zařízení jsou Panlab partners, kteří se také starají o jejich správu.

PTM – *Panlab Testbed Manager* – Komponenta systému, je nainstalována v každém tesbedu, který je součástí systému Panlab. Zajišťuje univerzální rozhraní pro komunikaci mezi jednotlivými odlišnými testbedy a Teagle. Toto rozhraní může být voláno jak z Teagle, tak ze strany jednotlivých *resources*. Dále zajišťuje směrování požadavků z Teagle na jednotlivé *resources*.

IGW – *Interconnection Gateway* – Komponenta systému, instalovaná na každém testbedu. Umožňuje směrování VCT mezi rozdílnými *partner sites*. Dále umožňuje uživatelům a *Panlab partners* vzdálený přístup k jejich VCT, pomocí tunelování.

Partner site – Doména Panlab partnera, je oddělená mrakem internetu od ostatních domén systému Panlab a ostatních partner sites.

5.2.1 Cílová skupina uživatelů

Systém Planetlab je určen pro široké spektrum uživatelů. Od velkých společností až po malé nebo střední podniky nebo výzkumné organizace.

- Malé a střední podniky jsou do systému zapojeny jako uživatelé, kteří zde testují své nové technologie a aplikace. Protože tyto podniky nemají jiný přístup k velkým testovacím prostředím, využívají možnosti systému Planetlab a poskytují dále zpětnou vazbu do systému.

- Velké firmy, jako např. výrobci síťových komponent se do systému přihlašují za účelem otestování svých výrobků a získání zpětné vazby od uživatelů.

- Výzkumná pracoviště se připojují za účelem nasazení svých výzkumných výsledků a zhodnocení funkčnosti na rozdílných platformách a simulačních prostředích.

5.2.2 Role v systému

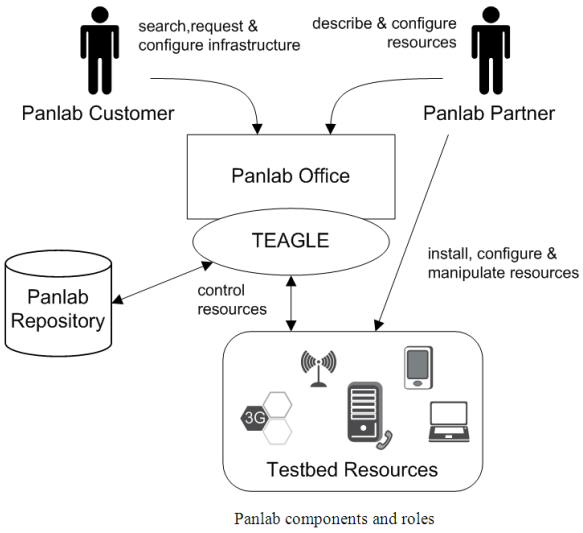
Infrastruktura systému Panlab spravuje spojení různých distribuovaných testbedů za účelem poskytování možnosti testování síťových technologií. Jako koordinační centrum zde působí *Panlab Office*. To zajišťuje veškerou správu a údržbu systému. V systému potom existují další role:

- **Panlab partner** – Poskytovatel fyzických infrastrukturních prvků potřebných pro provozování veškerých služeb. Všichni *Panlab partners* jsou připojeni k *Panlab Office,* přes kterou poskytují dále uživatelům své infrastrukturní prvky.
- **Panlab customer** – využívá služeb, které poskytuje *Panlab Office*. Připojuje se taktéž k *Panlab Office,* kde se potom dále může přihlašovat k jednotlivým testovacím prostředím.
- **Panlab Office** – Poskytuje zprostředkovatelskou službu mezi *Panlab partners* a *Customers*.

5.3 Architektura systému

Protože systém Panlab sdružuje více testbedů pod jedno vnější rozhraní, musí obsahovat množství architektonických komponent. Hlavním řídícím subjektem systému je portál Teagle. To je webové rozhraní, přes které uživatelé přistupují k systému. Každý zúčastněný testbed musí obsahovat *Panlab Testbed Manager Component PTM*. Tyto PTM mají přístup ke kolekci zdrojů,

která je kontrolována a spravována *Resouce Adapters* RA. RA umožňují všem zdrojům, aby byly vystaveny a nabízeny přes rozhraní *Teagle*. Toto rozhraní poskytuje ještě další prvky pro koordinaci celé federace. Poskytuje rozhraní pro správu poskytovatelům jednotlivých testbedů, dále nástroj pro vytváření *Virtual Customer Testbeds* VCT a úložný prostor *Panlab repository* kam se ukládají např. výstupy z jednotlivých testů. Při návrhu *Panlab repository* bylo potřeba oddělit klientské aplikace a služby pro přístup k datům a komunikaci. Obě dvě využívají protokol http. Pro toto oddělení byly zvoleny způsoby SDO a XML-RPC z důvodu velkého množství již existujících knihoven a možností implementace.



Obrázek 6: Znázornění vztahů v systému, převzato z Panlab.net

Přesné grafické znázornění uživatelů a základních prvků systému je znázorněno na obrázku 6. Jednotlivé prvky systému jsou popsány v kapitole Architektura systému.

5.3.1 Teagle

Architektura systému se opírá o základní Framework, který se jmenuje Teagle. Jedná se o webovou službu, která poskytuje zákazníkům rozhraní, pomocí kterého mohou komunikovat se systémem.

Architektura systému Panlab je celá založena na Frameworku *Teagle*. Jedná se o webovou aplikaci, která poskytuje rozhraní mezi zákazníky a systémem, sběrem jejich požadavků na tvorbu testbedů a možnostmi najít použitelné komponenty, pomocí kterých by se daly jejich požadavky realizovat. *Teagle* vlastně zajišťuje hledání a sestavování testovacích prostředí v databázích *Panlab partners*. To zahrnuje nezbytnou rezervaci požadovaných zdrojů (komponent) a jejich propojení pro možnost poskytnou každému zákazníkovi specifický testbed. Zákazníci mohou vybrat zvolené technologie a prvky s kterými budou chtít pracovat a Teagle zajistí jejich rezervaci, propojení a spuštění. Z důvodu používání různých technologií v rozdílných testbedech u různých *Panlab*

partners a rozdělení testbedů do různých administrativních domén je třeba implementovat tyto tři prvky:

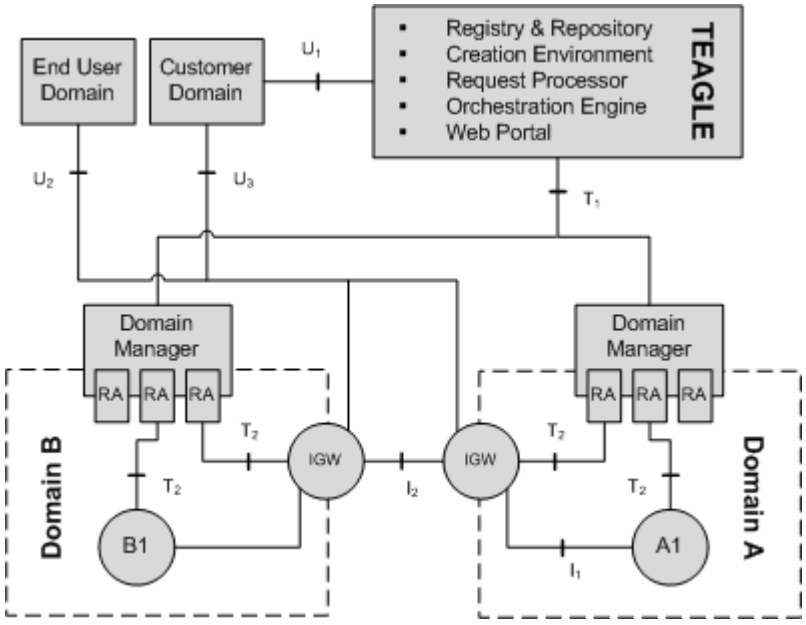
- Resource Description
- Service Exposure
- Service Composition.

Teagle portal - Pro přístup k *Teagle* se využívá *Teagle portal*, který umožňuje sestavovat *Virtual Customer Testbeds VCT*. Tyto testbedy jsou složeny z množství distribuovaných softwarových a hardwarových komponent, umístěných v existujících testbedech v rámci Evropy. VCT poskytují možnost vytvoření specifických virtuálních testbebů přesně podle specifikací klienta.

Teagle portal také funguje, jako centrální koordinační bod, který sdružuje všechny *Panlab partners* a jejich fyzické testbedy.

Pro popis zdrojů, je definovaný doporučený informační model. Ten umožňuje *Teagle* porozumět funkcionalitě, kterou poskytují zdroje a doporučit tak uživatelům odpovídající prvky. O to se stará komponenta PTM (*PanLab Testbed Manager*). Sestavení testbedů je založeno na *resource description* všech aktuálně dostupných komponent, spolu s instrukcemi, jak je obsluhovat.

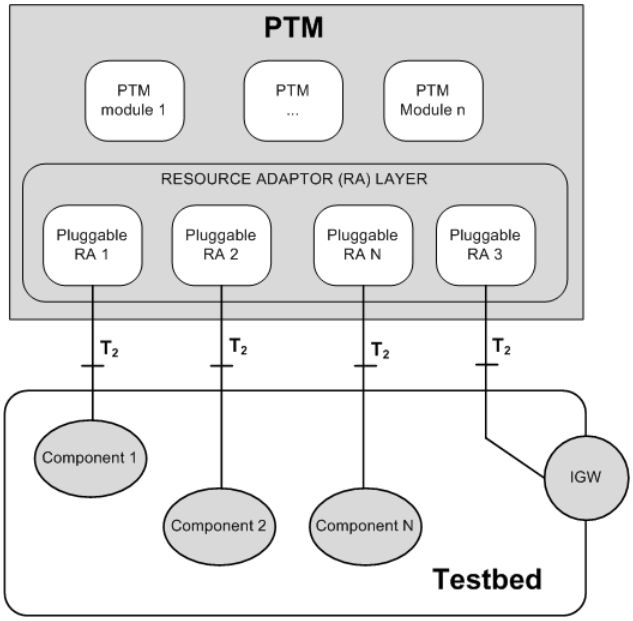
Následující obrázek č. 7 popisuje vztahy v systému. Hlavní komponentou je Teagle, to zajišťuje komunikaci s jednotlivými doménami *Panlab Partners*. Každá doména má své vlastní rozhraní – *Domain Manager* a dále jednotlivé RA, které zajišťují komunikaci přímo s jednotlivými *resources*. Dále je zde komponenta IGW, která zajišťuje komunikaci mezi jednotlivými testbedy. Všechny zmiňované komponenty budou popsány v následujících kapitolách.



Obrázek 7: Architektura systému, převzato z www.Panlab.net

5.3.2 PTM

Jedná se o implementaci interakce komponent na *controll* vrstvě[6] mezi požadavky a konfiguračními příkazy přicházejícími z *Teagle* a jednotlivými komponentami v testbedech. PTM je schopno překládat generické povely generované *Teagle* (např. Read, write, update, delete…) na příkazy specifické pro jednotlivé komponenty testbedů např. SNMP příkazy, které jsou proveditelné přímo v prostředí daného testbedu. PTM takto vytváří mapování mezi *federation level commands* (*Teagle*) a *specific commands*(pro každé specifické zařízení).



Obrázek 8: Interakce testbedu s PTM, převzato z www.Panlab.net

5.3.3 IGW

Interconnection Gateway. Jedná se o službu zodpovědnou za poskytování konektivity do jiných administrativních domén. Zajišťuje vzájemné propojování testbedů a jednotlivých komponent testbedů společně s *peers,* kteří jsou součástí VCT konfigurace. Technicky je tato služba hranicí, která zajišťuje komunikaci a vzájemnou autorizaci testbedů a jejich komponent. Pracuje na 2 vrstvě a zajišťuje implementaci následující funkcionality:

1.L2 konektivita na modelu ISO OSI nebo izolace jednotlivých testbedů a jejich komponent.
2. Služby VPN komunikace mezi jednotlivými komponentami.
3. Možnosti Firewallu a filtrování provozu.
4. Aplikace proxi a překladu adres.

[6] Vrstva systému, interní síť sloužící pro vzájemnou komunikaci uvnitř systému mezi jednotlivými komponentami při sestavování VCT. Není přístupný uživatelům, ti pracují na jiné vrstvě – uživatelské.

5.3.4 RAL

The Resource Adaptation Layer. Jedná se o koncept integrace zdrojů nabízených vlastníky jednotlivých testbedů (*Panlab Providers*) v rámci platformy Panlab. V současnosti je každá instance RAL založena na shromažďování aktuálně spuštěných *Resource Adaptors* (RA). RA poskytují univerzální rozhraní pro komunikaci směrem k PTM modulům, i když většina PTM modulů využívá jiné specifické komunikační protokoly a poskytují jiné možnosti konfigurace. RA vystupují jako drivery pro jednotlivé *resources,* rozdílných testbedů.

RAL je komunikační Framework používaný pro komunikaci mezi RA a centrálním PTM, které umožňuje vývojářům RA využívat jejich vlastní metodologii, pro tvorbu komunikačních protokolů. RA jsou registrovány u Frameworku PTM a dávají o sobě vědět zbytku celého systému. PMT modul slouží k sledování událostí, které probíhají na jednotlivých RA. Vzhledem k tomu, že RA jsou pouze referencí na skutečné fyzické zdroje (komponenty), musí také umět předávat zprávy o chybách nebo výpadcích fyzických zdrojů dál do systému. Děje se to např. při přerušení spojení, chybě komponenty nebo při nezbytných updatech aplikačního softwaru.

Každý *resource* je reprezentován odpovídajícím RA jako množina atributů. Pro dosažení co nejlepších výsledků komunikace a zároveň umožnění určité autonomie vlastníkům zdrojů se používá toto schéma.

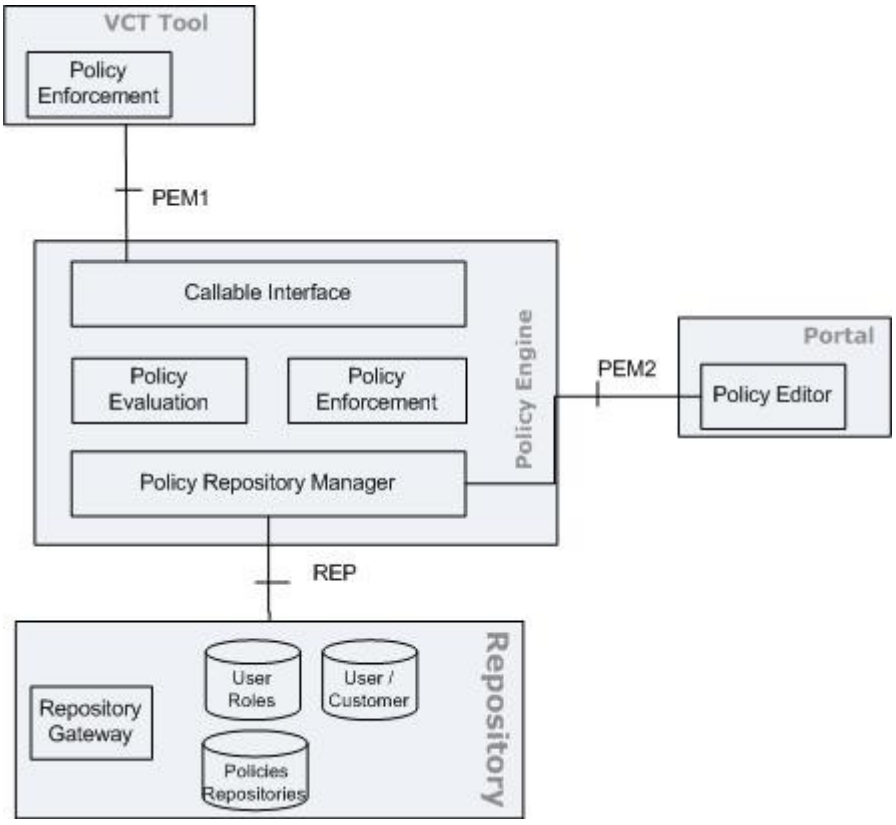
- *generic XML schema* slouží pro popis všech možných konfiguračních parametrů zdrojů. Všechny zdroje jsou popsány podle předem definovaného schématu pomocí XML dokumentu.
- Pro popis slouží atributy:jméno, hodnota, možnost editace a default hodnota

Při registraci u PTM se RA vývojář nebo vlastník zdrojů zaregistruje také XML dokument, který musí odpovídat určenému komunikačnímu schématu a tento dokument je potom připojen přímo k *Teagle* a slouží pro překlad komunikace. Obsažené informace musí odpovídat zařízením, pro které je toto překladové schéma navrženo.

5.3.5 Komponenta pro kontrolu bezpečnostních politik - Teagle Policy Engine

Teagle Policy Engine poskytuje možnost provozovatelům *resources* definovat pravidla a omezení týkající se použití jejich zařízení. Tyto politiky se dají chápat jako soubor povolených a zakázaných akcí a nastavení, které jsou pro dané uživatele nebo skupiny uživatelů povolené na zařízeních. V systému existuje mnoho stejných zařízení, které aplikují různé bezpečnostní politiky, tudíž jejich možnosti nastavení jsou různě omezené. Ve většině případů jsem se setkal s tím, že zařízení se striktními politikami byly daleko méně požadované a vytěžované, a naopak zařízení s volnějšími politikami nebo s téměř neomezeným přístupem byly naopak pořád v zabookované a měly čekací doby na rezervaci plné, i na několik týdnů dopředu.

V zájmu co největšího omezení časově a strojově náročných operací, jsou operace jako třeba rebookování nebo vracení změn nastavení zařízení, se tyto operace snaží systém omezit na minimum. Uživatel si proto celou topologii sestaví tzv. offline, potom je zkontrolována její kompatibilita se systémem a až po těchto úkonech, se teprve začnou uživatelské požadavky vykonávat na *resources*. Z mých zkušeností se systémem vyplývá, že to sice snižuje možnosti změn nastavení zařízení při běhu experimentu, ale zase to hodně usnadňuje režijní činnosti samotného systému a umožňuje zkrátit dobu experimentů na minimum. Některá dynamická nastavení zařízení se samozřejmě dají měnit i při běhu experimentu, např. změnit směrovací protokol nebo použít jiné adresování. Ale naopak se nedají přidávat další zařízení za běhu. To přináší výrazné zrychlení uvolňování zařízení jiným uživatelům. Další plus pro rychlost uvolňování *resources* je, že většina nastavení se děje tzv. offline. To znamená, že nastavení se provede pomocí nástroje VCT tool, potom je uloženo a zkontrolováno a až po potvrzení kontroly systémem je teprve umožněno kontaktovat *resources* a v případě, že jsou volné, začít nastavení nahrávat a spouštět experiment.



Obrázek 9: Policy Engine, převzato z www.Panlab.net

Teagle Policy Engine je postaven na specifikaci OMA Policy Evaluation, Enforcement and Management, viz:

http://www.openmobilealliance.org/Technical/Release_program/peem_v1_0.aspx

Jedná se o definici Frameworku pro kontrolu bezpečnostních politik na platformách poskytovatelů služeb. Díky využití objektového pohledu na politiky a použití *Drools rule langure*

pro definici politik, jsou bezpečnostní politiky v systému chápány jako balíčky, které se připojují k jednotlivým komponentám systému.

Na následujícím obrázku převzatém z www.panlab.net je vidět, že s *policy engine* spolupracují Komponenty *Teagle portal* a *VCT tool*. Pomocí *Teagle portal* mohou *panlab partners* definovat nové bezpečnostní politiky pro omezení přístupu k jejich *resources*. A na druhé straně nástroj *VCT tool*, který tyto politiky využívá a aplikuje jejich omezení, na koncové uživatele systému.

5.4 Softwarová architektura

V architektuře systému Panlab *Teagle* poskytuje skupinu nástrojů pro interakci s požadovanými *resources*. Aplikace obsažené v této skupině vyžadují sdílený ukládací prostor pro sdílení společných dat. Jako toto sdílené úložiště slouží *Panlab repository*.

Například pomocí nástroje *VCT* si uživatel může specifikovat testbed podle svých představ, pomocí jednoduchého drag-and-drop mechanizmu. Může takto propojit, nastavit a spustit všechna vybraná zařízení. *Teagle* potom automaticky sestaví *testbed setup* nad několika různými doménami *Panlab partners* a zde rezervuje a nastaví, požadované zdroje. *Repository* poskytuje přístup k souvisejícím datům z rozdílných zdrojů, od různých Panlab komponent a poskytuje je přes standardizované rozhraní.

Pro poskytování těchto služeb slouží *RESTful (REpresentation State Transfer)* architektura. V této architektuře je důraz kladen na rozdílnost zdrojů. Každý má svůj jedinečný identifikátor a je přístupný na jednoznačné URL adrese, takže vyhledávání a přístup k zařízením, je dokonale jednoznačný.

Softwarová architektura *repository* je složena z množství aplikací běžících na aplikačním serveru. Každá aplikace má své vlastní datové úložiště. Je to množina webově přístupných *RESTful* rozhraní za kterými se nacházejí *REST* zdroje. To ukazuje *repository* jako server v architektuře klient – server. Každá aplikace může také být klientem pro jakoukoliv jinou aplikaci pomocí poskytnutí opakovaného přístupu k vlastním *RESTful resources*. Tohle oddělení, od obsahu klientských aplikací, poskytuje možnost čistého přístupu k *repositury,* v rámci celé architektury systému. *Repository* zde slouží pouze pro ukládání a příjímání dat, od klientských aplikací. Ostatní požadavky, jako generování sestavovacích skriptů pro testbedy, změny stavů zdrojů (komponent) nebo správa *VCT* je předána specifickým aplikacím, které *repository* využívají pouze jako místo pro sdílení dat. To dovoluje vyvíjet tyto aplikace nezávisle na *repositury,* ale stále používat stejný datový model.

Komunikace s *repository* je bezstavová. Každý pár *request / responce* musí obsahovat všechna data, které bude potřebovat jak klient, tak server, pro zpracování dat. Stav *session* a obsah zpráv je kontrolován jednotlivými klientskými aplikacemi, což umožňuje poskytovat rozsáhlé množství bezchybných operací.

Díky tomu, že je ze strany architektury podporováno cachování, je ušetřena velká část zátěže na straně serveru (*repository*). Díky tomu, že jsou požadavky ukládány do mezi paměti a odsud znova načítány, je ušetřeno mnoho času, který by jinak byl potřeba k znovu nalezení a shromáždění požadovaných informací.

To vede k použití systému vrstev, kde každá entita může pracovat pouze na úrovni své vrstvy. To umožňuje implementovat architekturu *repository* přesně tak, jak bylo výše popsáno, včetně cachování a load balancingu, bez nutnosti modifikovat klientské aplikace. Tato architektura umožňuje použít jakýkoliv datový model. Současně používaný datový model popsaný v následující kapitole je proto pořád ve fázi testování.

5.4.1 Datový model Panlab repository

Veškeré testbedy poskytované *Panlab partners* mají jednu společnou vlastnost. Jsou složeny z různých *resources*, které se nacházejí distribuované v různých umístěních. Každá komponenta požaduje jiné vstupní atributy a naopak poskytuje jiné metody a funkcionalitu. To vyžaduje nutnost, množství rozdílných rozhraní poskytovaných těmito komponentami zastřešit pod jedno sdílené rozhraní.

Pro automatizaci sestavování uživatelských testbedů se používá *Teagle,* ale to také potřebuje možnost, jak se všemi rozdílnými komponentami komunikovat a spravovat je.

Pro tyto potřeby byl vytvořen informační model, který umožňuje všechny různorodé komponenty začlenit do systému. Tento model musí umět popsat jak vztahy a možnosti propojení jednotlivých komponent, tak i možnost funkcionality jednotlivých komponent.

Informační model je v systému Panlab definován jako „*representation of the characteristics and behaviour of a component or system independent of vendor, platform, language, and repository"* (Strassner et al. 2007) - reprezentace vlastností a chování komponent nezávislé na poskytovateli, platformě, jazyku a úložišti. Informační model slouží jako abstraktní popis fyzických komponent, jak síťových zařízení, tak uživatelů a dalších entit v systému a vztahů mezi nimi.

Pro sestavování automatických sítí v systému se používá model *DEN-ng*. Jedná se o UML model, který popisuje síťové prvky jako entity, jejich charakteristiky a vztahy, dále k nim doplňuje možnost nastavení životního cyklu. Tento model byl zvolen jako vzor pro *Panlab Data model,* kvůli třem hlavním kritériím.

- Vzor byl od začátku koncipován pro maximální podporu návrhových vzorů, hlavně vzoru *role pattern*. Hlavně kvůli možnosti vlastní rozšiřitelnosti.
- Používá velké množství abstrakcí, které hodně zjednodušují správu systému.
- Jedná se o jediný model, který podporuje organizaci objektů během změn jejich životních cyklů. Model *DEN-ng* pro modelování životního stavu objektů používá stavových strojů.

Díky použití *DEN-ng* modelu umožňuje systém všem poskytovatelům testbedů zadávat zařízení do systému jako fyzická nebo logická, což umožňuje daleko lepší správu zdrojů. Tato organizace zdrojů umožňuje provádět lepší nastavení a monitoring zdrojům za účelem dosažení

lepších výkonů při sestavování uživatelských testbedů. Každý uživatel si svůj testbed ukládá do třídy *VCT*. Může jej sdílet s dalšími uživateli a měnit mu jednotlivá nastavení. Detailní třídní diagram popisující PanLab Data Model je uveden na obrázku 10.

5.4.2 Implementace

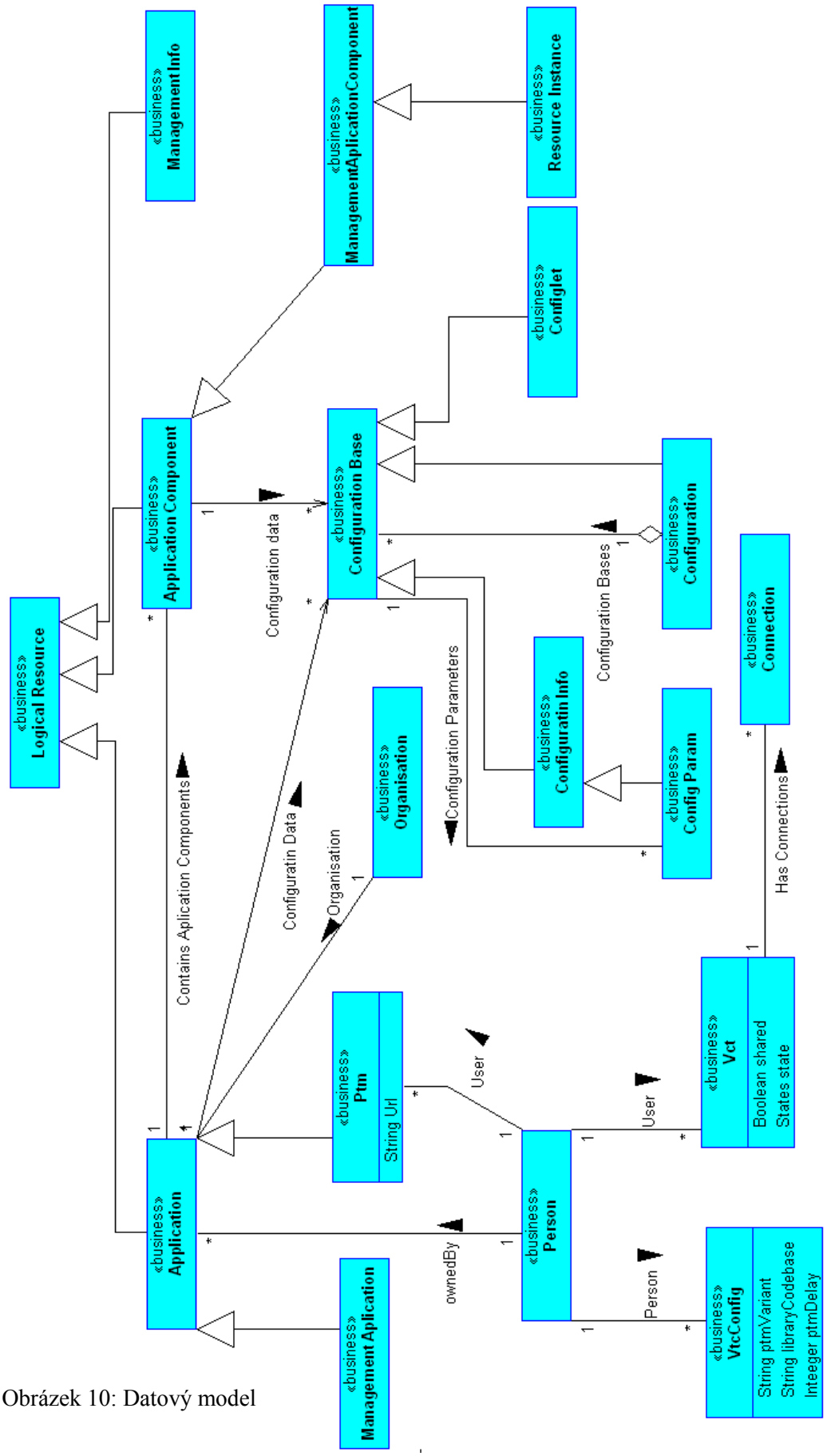
Jak bylo popsáno v předchozí kapitole, *RESTfull* architektura poskytuje jednotný, univerzální přístup k *resources.* Toto jednotné rozhraní umožňuje přístup k datům i pro jiné aplikace, bez potřeby jakéhokoliv speciálního kódu. Stačí zde pouze standardní metody HTML jako GET, POST, PUT a DELETE. To je obzvlášť přínosné, protože *testbed repository* může být integrováno do mnoha rozdílných aplikací v rámci celého systému Panlab. Například to jsou *Teagle portal, VCT tool* a *PTM*. To bylo také velkým přínosem při vývoji, protože ne všechny požadavky byly hned úplně jasné a byla vyžadována určitá míra flexibility.

Pro implementaci *repository* aplikací byl zvolen Framework *Grails*. Jedná se o open source web application framework, který využívá jazyk *Groovy* a *Java web development*. Framework obsahuje mnoho zařízení, která jsou využívána pro vývoj *repository* aplikací. Zaprvé, jedná se o MVC Framework, který odděluje aplikační doménový model od uživatelského rozhraní (*view*) a aplikační logiky (*business logic*). To umožňuje zaměřit se předně na vývoj doménového modelu, který je součástí implementace *Panlab data model*.

Framework umožňuje implementaci *REST resources* ve formě URL mapování na metody *controll* vrstvy. Výchozí mapování ve *Grail* je `/controller/action/id`. To může být samozřejmě změněno. Dále se dají nastavit (GET, POST, PUT a DELETE) na ekvivalenty pro REST v `UrlMappings.groovy`.

Ukázka jednoduchého GET požadavku na proměnnou *person,* z *repository aplication.*

```
GET /repository/rest/person/1
<?xml version="1.0" encoding="UTF-8"?>
      <person id="4587">
            <personRoles>
                  <personRole id="3"/>
            </personRoles>
            <userName>drihosek</userName>
            <fullName>David Rihosek</fullName>
            <emails>
                  <email id="1"/>
            </emails>
      <password>5gs0f33516b8276d758a9a7274737417</password>
      <organisations>
            <organisation id="547"/>
      </organisations>
</person>
```



Obrázek 10: Datový model

Framework obsahuje mnoho podpůrných mechanizmů pro vývoj aplikací. Při implementaci *repository* aplikací se ukázala jako vynikající možnost mapování objektového modelu do vlastního relačního modelu. Framework je postaven nad *Hibernate 2010* a využívá knihovnu *object relational mapping (ORM)* pro vytváření a práci s *ORM called Grails Object Relational Mapper (GORM)*. Výhoda použití jazyka *GROM* spočívá v automatickém mapování tříd do databáze. Kde hodně *OMR* objektů vyžaduje mapování pro následný externí přístup. Tato výhoda umožňuje implementátorům soustředit se přímo na implementaci aplikační logiky a ne na mapování tříd do DB.

Pro generování uživatelského rozhraní se používá *scaffolding and templating* mechanizmus. *Scaffolding* generuje většinu operací typu read/write automaticky a nevyžaduje větší implementační zásahy, což také znamená značnou úsporu času.

5.5 Práce se systémem

V následující kapitole budou popsány základní kroky, jak obsluhovat systém. Veškerá zde uvedená funkcionalita je osobně odzkoušená. Jednotlivé podkapitoly jsou doplněny osobními zkušenostmi s používáním systému.

5.5.1 Registrace a přihlášení do systému

Registrace je jednoduchá, stačí vyplnit pouze registrační informace na stránce http://www.fire-teagle.org/create_account.jsp. Jedná se o uživatelské jméno, heslo, emailovou adresu a o organizaci, ke které uživatel patří. V mém případě jsem zvolil volbu organizace *Other*. Po potvrzení a odeslání registračního formuláře systém vygeneruje potvrzovací email a pošle jej na zadanou adresu. V mém případě jsem na potvrzení registrace a povolení vstupu do systému čekal několik týdnů. V emailu byl také dotaz, na jaké účely hodlám systém využívat, odpověděl jsem, že pro akademické a studijní účely. Na základě těchto informací mě byl následně vygenerován účet v systému.

Pro přístup k systému se používá portál *Teagle*, přihlášení je dostupné na adrese http://www.fire-teagle.org. Obrázky z průběhu přihlašování a registrace jsou dostupné v příloze v elektronické podobě.

5.5.2 Teagle VCT tool

Jedná se o součást portálu *Teagle*. Je to webová aplikace, přes kterou si uživatelé sestavují požadované VCT, experimenty z dostupných zařízení.

5.5.2.1 Implementace

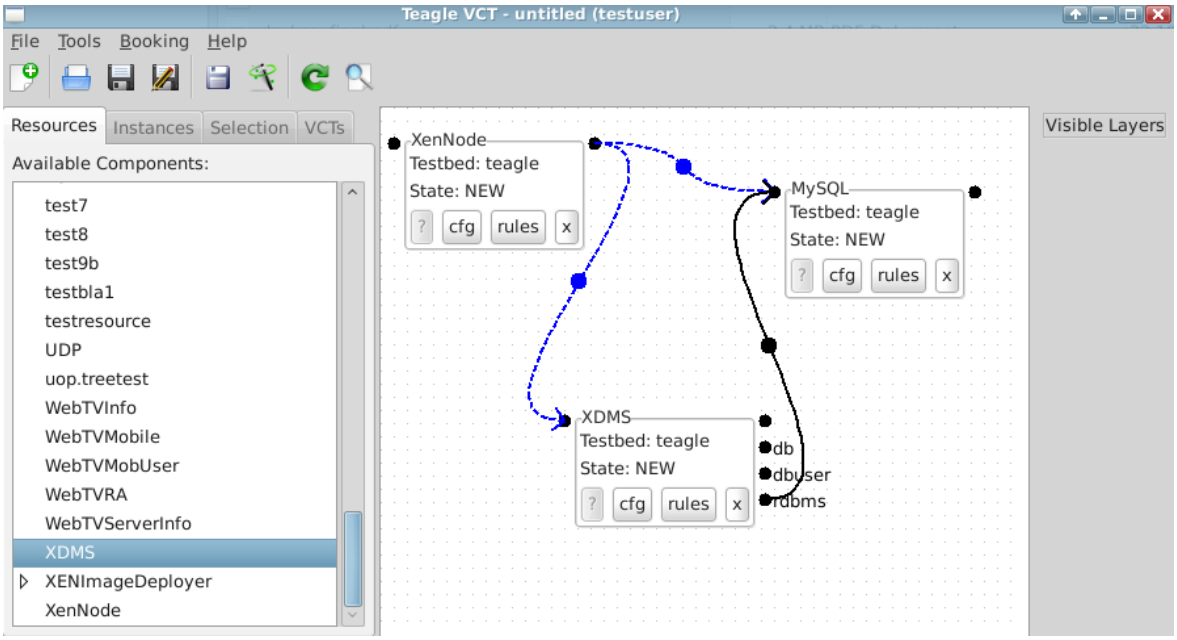
VCT tool je implementováno jako Java aplikace, která se spouští přes *Java Webstart launcher* dostupný ve standardní instalaci prostředí Java. Díky použití tohoto webového řešení

přístupu, odpadá nutnost instalovat jakýkoliv software do klientského počítače a umožňuje přístup k systému téměř odkudkoli.

VCT tool při staru i při běhu využívá *TeagleRepository*, protože jsou zde poněkud striktní politiky ohledně používání zařízení a jedná se o přenos většího množství dat, je lepší pracovat s rychlejším připojením. V mém případě jsem pro testování použil připojení rychlosti 4 Mbit, a odezvy systému byly poněkud pomalejší. Z *Teagle Repository* si *VCT tool* stahuje informace o předchozích používaných testbedech a aktuálně volných zařízeních. Z důvodu pružnějšího zobrazování informací se dá zvolit, kterou *repository* primárně využívat pro zobrazování. *VCT tool* využívá zobrazování temporary dat v průběhu načítání aktuálních požadavků. Z tohoto důvodu existují dvě repository: Interní stažená a uložená v Tempomary files u uživatele a aktuální verze na webu. Při práci, se samozřejmě interní *repository* aktualizuje. *VCT tool* dále umožňuje ukládat uživatelská nastavení a samozřejmě aktuálně rozpracované projekty, dá se k nim poté přistupovat přes webové rozhraní.

5.5.2.2 Spuštění VCT tool

Pro spuštění je potřeba být registrován v systému a přihlásit se k portálu *Teagle*. Zde se VCT spouští z sekce: *Members area* / *VCT Design*.



Obrázek 11: VCT Tool

5.5.2.3 Práce s VCT tool

GUI aplikace se skládá z několika částí. Hlavní částí je *main grid* (hlavní mřížka), kde se zobrazují a konfigurují aktuálně ovládaná zařízení. Dále jsou zde panely, ve kterých se prochází

dostupná zařízení, jejich konfigurace a ověřuje se dostupnost atd. GUI je zobrazeno na obrázku x, kde je vidět i sestavování jednoduchého VCT. Na příkladu z obrázku klientský počítač komunikuje s vzdálenou SQL databází a XML Data Management Serverem. Všechny jsou součástí defaultního testbedu *Teagle*.

Dále jsou zde vlevo 4 záložky:

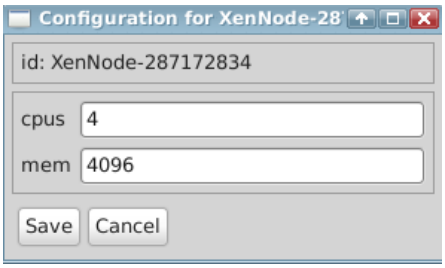
- **Resources** - Seznam *resources*, zdrojů se kterými se dá pracovat, dají se vybírat a připojovat do aktuálního projektu. Výběrem určitého ressource se rozumí vytvořit si jeho novou instanci a tu si přidat do vlastního projektu.
- **Instances** - Obsahuje již vytvořené instance *resources*, i z jiných dřívějších projektů uživatele. Velice dobře to umožňuje používat již jednou vytvořená a nakonfigurovaná zařízení i v jiných budoucích projektech.
- **Selection** – vyhledávání instancí *resources* mezi dalšími uživateli. Pokud například někdo již nakonfiguroval směrovač pro použití, dá se použít jeho konfigurace a pouze upravit některé detaily a použít již vytvořenou instanci.

Dále program obsahuje menu, kde zajímavá je záložka *Booking*. Záložky *File, Tools* a *Help* jsou standardní nabídky používané v mnoha jiných programech.

Záložka *Booking* slouží k bookování si požadovaných zařízení v systému. Protože většinu času jsou všechna zařízení využita, při tvorbě většího experimentu je třeba si zařízení už několik dní dopředu zamluvit. Přesný postup bookování zařízení bude popsán v další kapitole.

5.5.2.4 Tvorba vlastního experimentu

Hlavním úkolem nástroje *VCT tool* je umožnit uživatelům sestavovat si vlastní testovací prostředí složené z jednoho nebo více *resources* nabízených *Panlab partners*. Tyto *resources* jsou hlavními entitami, se kterými *VCT tool* pracuje. Při staru se začíná s prázdným experimentem – VCT. Na začátku je potřeba si zabookovat zvolená zařízení a přidat jejich instance do *main grid*. Některé *resources* mohou být dostupné z více domén, to znamená, že stejný typ zařízení do systému připojili různí *Panlab partners*. V tomto případě *VCT tool* nabízí výběr zařízení a domény, ze které se má ressource zvolit. Již instanciované *resources* se dají nastavovat pomocí tlačítka *CFG*. Na obrázku číslo 12 je ukázka konfigurace klientského počítače, dá se nastavit počet procesorů a velikost paměti.



Obrázek 12: Konfigurace PC

Konfigurace různých zařízení je různá, záleží na typu zařízení. Například směrovače nebo přepínače mají mnohem složitější a komplexnější možnosti nastavení. Na všech zařízeních se dají nastavovat pouze ty funkce, které povolí *Panlab partner,* který dané zařízení vlastní. Složitější síťová zařízení se nenastavují přes dialogová okna jako počítač, na obrázku ale mají povolen přímý vzdálený přístup přes konzoli. Veškerá komunikace se však odehrává přes rozhraní *VCT tool*. To jestli však bude k zařízení poskytnut neomezený nebo pouze omezený přístup, rozhoduje jeho vlastník *Panlab partner*. Je tedy celkem normální, že dvě stejná zařízení budou mít naprosto jiný přístup k nastavení. V mém případě se mi podařilo zabookovat si dva stejné směrovače, kde na jednom byl umožněn přístup přes konzoli a možnost plného nastavení, kdežto na druhém se dal nastavit pouze použitý směrovací protokol a IP adresy rozhraní. Veškerá další funkcionalita byla omezena. Zde se nastavení provádělo podobně jako na ukázkovém obrázku 12 pomocí dialogového okna.

Dalším nevyhnutným omezením je nutnost používat celé doporučené sady zařízení, která se pro bezchybnou funkci navzájem vyžadují. Tyto vzájemné vazby jsou taktéž definovány *Panlab Partners*. Například pokud by uživatel požadoval Asterisk server pro testování, *PTM* automaticky instaluje i nový MySQL server nebo nakonfiguruje a nasdílí již použitý a další požadovaná závislá zařízení. Pro definici těchto vzájemných závislostí zařízení se používá dvě odlišné metody spojení.

Containment – Znamená, že jedno *resource* žije uvnitř jiného. Například na jednom serveru mohou být nainstalované různé služby. Přidáním tohoto serveru do experimentu se zároveň přidají veškeré podporované služby. Vazba by se dala připodobnit k vazbě kompozice v jazyce UML.

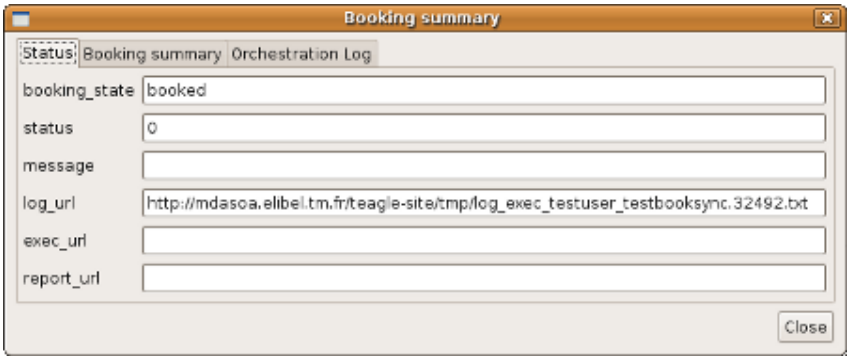
References – Znamená, že jedno *resource* využívá jiné. Například takhle mohou být navzájem svázány Aplikační a databázový server aplikace. Samotná aplikační vrstva by bez databáze byla k ničemu a naopak. Vazba by se dala připodobnit k vazbě agregace v jazyce UML.

5.5.2.5 Bookování resources

Jakmile uživatel dokončí sestavení svého experimentu v *VCT tool*, musí kliknout na tlačítko *Book*. Následně mu bude zobrazen seznam všech požadovaných *resources*. Po potvrzení systém odešle XML dokument se seznamem zařízení *Orchestration Engine*, který se stará o rezervaci zařízení. Ten dále kontaktuje jednotlivá zařízení a zajišťuje distribuci požadavků k požadovaným *resources*. Jakmile je proces rezervace zařízení dokončen, systém zobrazí okno potvrzení rezervace viz. obr 13.

Pokud se proces rezervace zařízení nezdaří, je třeba opravit chybné informace a proces rezervace opakovat. Systém bohužel neumí rezervovat pouze některá zařízení a ty ostatní po úpravě parametrů postupně doplnit. Typické chyby při sestavování topologie jsou například: Pokud jsou na zařízeních ponechány defaultní IP adresy 0.0.0.0 nebo pokud se uživatel pomocí *VCT tool* snaží propojit vzájemně nekompatibilní zařízení nebo pokud v konfiguraci nějaká zařízení chybí. Např. pokud uživatel požaduje MsSql server, ale nemá v projektu přiřazen žádný fyzický server, na kterém by mohl tento běžet…

Před restartem procesu rezervace je dobré prohlédnout si *Orchestration Log* v dialogu *Booking summary* a podívat se na čem proces havaroval. Jakmile jsou nedostatky v nastavení odstraněny, je možné spustit proces rezervace znovu.



Obrázek 13: Booking summary

V obou případech, ať už rezervace proběhne úspěšně či nikoli, na konci se všechny změny provedené v systému projeví i ve *VCT tool* a uživatel na ně může pružně reagovat. (Jsou změněny stavy *rources* a umožněna práce s nimi). Stav, ze kterého se dá zařízení rezervovat, je pouze stav *unbooked*. Zařízení je ve všech jiných stavech pro koncové uživatele nedostupné.

5.5.2.6 Vyhledávání resources

Protože v systému Panlab je zapojeno mnoho zařízení, jejich procházení může být zdlouhavé. Proto existuje funkce *Selection,* pomocí které se dají požadovaná zařízení ze seznamu vyhledávat. Dá se vyhledávat podle *Panlab partner,* který poskytl dané zařízení nebo podle dostupnosti.

5.6 Závěr

V projektu Panlab byl realizován jedinečný nápad spojení více rozdílných testbedů do jednoho systému a poskytnout tak uživatelům daleko lepší možnosti konfigurace, než by byly jednotlivé testbedy schopny poskytnout samostatně. Je tedy možné sestavit si vlastní testbed z resources obsažených v různých fyzických testbedech.

Protože se systém Panlab skládá z mnoha různých komponent distribuovaných do mnoha různých umístění, musí implementovat dokonalé nástroje a technologie pro synchronizaci komunikace a určovat jednoduché univerzální rozhraní pro přístup k těmto komponentám. Toto systém realizuje pomocí portálu Teagle a dalších na pohled uživatelům skrytým, ale velice důležitých součástí jako jsou např. *Panlab repository*.

Systém Panlab je si se systémem Virtlab velice podobný, pracuje také na podobných principech sdílení zdrojů a následném sestavování uživatelem definovaných testbedů, rezervaci komponent a tvorbě uživatelských konfigurací systému. Oba systémy poskytují uživatelům

možnost sestavit si vlastní síťovou topologii a nad ní potom provádět různá testování. U systému Panlab je vynikající možnost rozšiřitelnosti a univerzálnosti systému díky použitým technologiím. Hlavně díky vícevrstvé architektuře a důmyslně použitému systému dědičnosti a návrhových vzorů přímo v návrhu systému.

6 Perspektivní rozšíření pro systém Virtlab

Všechny tři testbedy popsané v diplomové práci poskytují různé možnosti testování síťových technologií. Systémy Emulab a PlanetLab jsou zaměřeny spíše na koncové stanice, kdežto systém Panlab, velice podobný systému Virtlab, poskytuje možnost sestavování vlastních testbedů z rozlišných zařízení.

Jednou z možností rozšíření by mohlo být, podobně jako v systémech Emulab a Planetlab, přidat možnost instalovat a spouštět na koncových uzlech vlastní software. Na druhou stranu, systém Virtlab neobsahuje dostatečný počet zařízení, ani neposkytuje takové možnosti v nastavení chování linek, aby byla tato funkcionalita použitelná. Protože testované testbedy pracují řádově nad stovkami zařízení, rozprostřenými na velké ploše, dají se zde s úspěchem testovat možnosti nastavení a chování aplikací v prostředí internetu.

Systém Virtlab je však primárně určen hlavně na jinou oblast použití. V tomto je podobný se systémem Panlab. Protože systém Panlab využívá mnoho uživatelů, je zde velice dobře rozpracován systém rezervací zařízení. Existuje zde také velice dobrá utilita, která je součástí obslužného softwaru VCT Tool, která kontroluje nastavení zařízení v uživatelském experimentu ještě před tím, než bude experiment nahrán přímo na fyzická zařízení, kde se potom bude vykonávat. Díky této funkcionalitě se šetří aktivní čas zařízení, protože ty pracují pouze nad topologiemi, které jsou zkontrolované a které by teoreticky měly fungovat. Praktická funkčnost se samozřejmě ověří, až na fyzických zařízeních. Tato utilita kontroluje zejména platnosti rozsahů IP adres a směrování, dále vyhledává konflikty nebo chybějící zařízení a ověřuje správnost základního nastavení. Např. nejde použít dvě stejné adresy v jedné podsíti, nechat defaultní adresy 0.0.0.0 nebo používat v názvu zařízení *hostname* mezery. Takováto funkcionalita sice vyžaduje velké implementační úsilí a hodně možností na ověřování, ale na druhé straně poskytuje velikou úsporu systémových prostředků. Systémovými prostředky není plýtváno zbytečně tam, kde jsou v zadaných topologiích základní chyby. Systém Panlab kontroluje správnost topologie ještě před tím, než uživatel provede rezervaci zařízení na určitou dobu. Pokud není topologie v pořádku, nedovolí systém uživateli vůbec zarezervovat si zařízení. V prostředí Virtlabu by to mohla být vhodná funkcionalita, která by pomohla rozmělnit nárazovou zátěž a omezila by počty přístupů studentů z důvodu špatných topologií a oprav až za běhu experimentu.

Předchozí možnost rozšíření však vyžaduje další funkcionalitu. A to podporu ukládání uživatelských topologií do nějakého univerzálního datového formátu. V praxi, u testovaných testbedů, ale i v jiných oblastech se dobře osvědčil univerzální formát XML. Tyto soubory by si mohl uživatel buď vytvářet a editovat ručně, nebo později by se mohl použít software k tomu určený. Testované testbedy obsahovaly dodávaný software, pomocí kterého se daly požadované topologie sestavovat. Velice to urychlovalo práci se systémem. Systém Virtlab je však koncipován tak, aby si v něm studenti procvičili syntaxi používanou v zařízeních Cisco. Proto by tento software

mohl podporovat jak grafické tak textové rozhraní. V grafickém rozhraní by se dalo pracovat klasickým způsobem *drag and drop* a pouze vyplňovat požadované parametry. Naopak textový režim by se tvářil jako konzola daného zařízení. Bylo by zde potřeba příkazy zapsat ručně.

Většina těchto funkcionalit používaných v testovaných testbedech je koncipována tak, že se systémy budou pracovat řádově tisíce uživatelů na velkých počtech zařízení. Proto jsou možnosti zobrazení a nastavení povoleny zadávat pomocí specifického software. Pro potřeby studentů a jejich testování si syntaxe příkazů a základního chování zařízení je však daleko vhodnější způsob zvolený v systému Virtlab. Tj. zjednodušeně řečeno, vzdálené přihlášení k zařízení a ovládání jej na dálku on-line a ne pomocí předem připraveného skriptu.

7 Závěr

V mé diplomové práci byly prozkoumány a zdokumentovány tři testbedy. Jednalo se o systémy Emulab, PlanetLab a Panlab. Každý z těchto testbedů nabízí poněkud jiný pohled na vytváření simulačního prostředí a také je určen pro jinou cílovou skupinu uživatelů. Systémy Emulab a PlanetLab jsou určeny spíše pro testování nových síťových technologií a jsou založeny na použití fyzických počítačů, které jsou podle simulačního skriptu nebo pomocí příkazů z konzole propojovány do požadovaných sítí a dají se na nich dobře testovat nové síťové technologie. Jsou tedy zaměřeny hlavně na koncové stanice a komunikaci mezi nimi. V systému Emulab se na každém node obsaženém v uživatelském experimentu spouští požadovaný obraz operačního systému a v něm může uživatel pracovat. V systému PlanetLab se naopak na každý fyzický počítač dá nasadit více virtuálních strojů, které mohou pracovat nezávisle na sobě a dělit se o fyzický výkon stroje.

Naopak systém Panlab je testbed založený na propojování více testbedů dohromady a sestavování uživatelem požadovaného testbedu z různých distribuovaných zařízení. Uživatel má možnost vybrat si dostupná zařízení ze seznamu, ty následně propojit a pomocí vzdáleného připojení nakonfigurovat jejich parametry a nechat na této topologii proběhnout svůj experiment. Součástí tohoto testbedu nejsou pouze počítače – koncové stanice, ale i další síťové prvky jako směrovače a přepínače různých výrobců. Díky univerzálnosti systému, kdy je veškeré propojování a nastavování zařízení řešeno přes portál Teagle, může systém nabídnout univerzální rozhraní, pomocí kterého se dá k různým síťovým prvkům přistupovat. Jakmile je veškeré nastavení provedeno a zařízení nakonfigurována, dají se přes portál Teagle spouštět vlastní programy a generovat provoz do sestavené sítě. Po dokončení experimentu jsou opět zařízení uvolněna a dále nabízena v systému pro další uživatele. Systém tak poskytuje věrnou simulaci chování sítí složených z reálných prvků.

Na všech testbedech jsem provedl malý experiment a otestoval deklarovanou funkčnost a použitelnost systémů. Všechny experimenty jsou uloženy v příloze diplomové práce na CD. Všechny soubory s experimenty obsahují komentáře, které usnadňují porozumění toho, co dané simulace provádí. Všechny testy byly zaměřeny spíše na otestování funkcionality a možností systému, ne přímo na testování nějaké technologie nebo protokolu. Přiložené skripty mohou pomoci budoucím uživatelům těchto systémů při seznámení se s prostředím a vytvořením složitějších testovacích topologií a testovacích případů.

Tato diplomová práce částečně navazuje na mou Bakalářskou práci Simulátory počítačových sítí. Systém Emulab totiž pro zadávání vstupních dat využívá TCL skripty síťového simulátoru NS-2, proto se zde na svou bakalářskou práci, kde je syntaxe a použití tohoto simulátoru analyzována, odkazuji.

Tato práce by také mohla sloužit jako námět pro budoucí implementaci nové funkcionality do systémů vyvíjených na VŠB. Detailní rozbor architektury testovaných systémů může být přínosem a inspirací pro budoucí tvůrce školních systémů. Dalším přínosem je prozkoumání a

osvětlení možností použití těchto systémů pro další použití. Budoucí uživatelé budou mít usnadněnu práci a po pročtení mé diplomové práce, mohou se plně věnovat tvorbě svých experimentů a neztrácet čas nad tím, který systém zvolit, jak funguje a jak se s ním pracuje.

Literatura

1. Emulab - Network Emulation Testbed Home [Online] Září 2010.
< *http://www.emulab.net/*>.

2. Google skupina - Emulab [Online] 2011.
[< http://groups.google.com/group/emulab-users?pli=1>.

3. Large-scale Virtualization in the Emulab Network Testbed. [Online] 2009.
< http://www.cs.utah.edu/flux/papers/virt-usenix08-base.html>.

4. An open platform for developing, deployng andaccesing planetary-lab services [Online]
< http://www.planet-lab.org/>.

5. PlanetLab presentatons [Online] 2011.
< http://www.planet-lab.org/presentations>.

6. Experiences Implementing PlanetLab [Online] 2006.
< http://nsg.cs.princeton.edu/publication/experiences_osdi_06.pdf>.

7. Securing the PlanetLab Distributed Testbed. [Online] 2004.
<http://www.usenix.org/event/lisa04/tech/full_papers/brett/brett.pdf>.

8. *Testbed FEDERICA* [Online] 2009.
< http://www.fp7-federica.eu/about/related.php>.

9. Panlab PanEuropean Laboratory Infrastructure Implementation [Online]2011.
<http://www.panlab.net/>.

10. PII Flyer . [Online] Září 2010.
<http://www.panlab.net/fileadmin/documents/Publications/02_PII_Flyer.pdf>.

11. Panlab Testbed Repository. [Online] 2010.
<http://www.panlab.net/testbed-repository.html>.

12. Strassner, J., Lehtihet, E. & Agoulmine, N. (2007), `Focale - a novel autonomic computing architecture:extended version', ITSSA journal . 14. Únor 2010. [Citace: 1 Březen 2011.]

Adresářová struktura přiloženého CD s přílohami

/Emulab	Systém Emulab
/PlanetLab	Systém PlanetLab
/Panlab	Systém Panlab

Každá složka věnovaná jednotlivému testbedu je dále členěna na pod složky obsahující spustitelné skripty, screenshoty z obrazovek a další doplňující materiály týkající se jednotlivých systémů.